

# **OpenVOS STREAMS TCP/IP Administrator's Guide**

Stratus Technologies  
R419-17

---

# Notice

The information contained in this document is subject to change without notice.

UNLESS EXPRESSLY SET FORTH IN A WRITTEN AGREEMENT SIGNED BY AN AUTHORIZED REPRESENTATIVE OF STRATUS TECHNOLOGIES, STRATUS MAKES NO WARRANTY OR REPRESENTATION OF ANY KIND WITH RESPECT TO THE INFORMATION CONTAINED HEREIN, INCLUDING WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PURPOSE. Stratus Technologies assumes no responsibility or obligation of any kind for any errors contained herein or in connection with the furnishing, performance, or use of this document.

Software described in Stratus documents (a) is the property of Stratus Technologies Ireland, Ltd. or the third party, (b) is furnished only under license, and (c) may be copied or used only as expressly permitted under the terms of the license.

Stratus documentation describes all supported features of the user interfaces and the application programming interfaces (API) developed by Stratus. Any undocumented features of these interfaces are intended solely for use by Stratus personnel and are subject to change without warning.

This document is protected by copyright. All rights are reserved. Stratus Technologies grants you limited permission to download and print a reasonable number of copies of this document (or any portions thereof), without change, for your internal use only, provided you retain all copyright notices and other restrictive legends and/or notices appearing in the copied document.

Stratus, the Stratus logo, ftServer, the ftServer logo, Continuum, StrataLINK, and StrataNET are registered trademarks of Stratus Technologies Ireland, Ltd.

The Stratus Technologies logo, the Continuum logo, the Stratus 24 x 7 logo, ActiveService, Automated Uptime, ftScalable, and ftMessaging are trademarks of Stratus Technologies Ireland, Ltd.

RSN is a trademark of Lucent Technologies, Inc.

All other trademarks are the property of their respective owners.

TCP Wrappers copyright information:

Copyright (c) 1987 Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Copyright 1995 by Wietse Venema. All rights reserved. Some individual files may be covered by other copyrights.

This material was originally written and compiled by Wietse Venema at Eindhoven University of Technology, The Netherlands, in 1990, 1991, 1992, 1993, 1994 and 1995.

Redistribution and use in source and binary forms are permitted provided that this entire copyright notice is duplicated in all such copies.

This software is provided "as is" and without any expressed or implied warranties, including, without limitation, the implied warranties of merchantability and fitness for any particular purpose.

NOTICE: Portions copyright RFC Editor, 1981-2006, (RFC 792, RFC 950, RFC 1256, RFC 1393, RFC 1475, RFC 2521, RFC 2863, RFC 3635, RFC 4022, RFC 4133, RFC 4293)

Manual Name: *OpenVOS STREAMS TCP/IP Administrator's Guide*

Part Number: R419

Revision Number: 17

OpenVOS Release Number: 19.3.1

Publication Date: January 2022

Stratus Technologies, Inc.

5 Mill and Main Place, Suite 500

Maynard, Massachusetts 01754-2660

© 2022 Stratus Technologies Ireland, Ltd. All rights reserved.

---

# Contents

---

<b>Preface</b>	xv
----------------	----

---

<b>1. Overview of STREAMS TCP/IP (STCP)</b>	1-1
Key Features	1-1
How to Find STCP-Specific Information	1-4
Components of STCP	1-5
Related Drivers	1-15
SDLMUX	1-15
Lower-Level Device Drivers	1-16
Hardware Requirements	1-17
Software Requirements and Considerations	1-17

---

<b>2. Configuring STCP, OSPFD, and STCP TELNET</b>	2-1
Checklist for Configuring STCP	2-1
Checklist for Configuring the TELNET Daemon	2-5
Checklist for Configuring the OSPFD Daemon Process	2-7
Checklist for TCP Wrappers Functionality	2-7
Checklist for Enabling IPv6 Support	2-9

---

<b>3. Configuring SDLMUX</b>	3-1
Overview of Device Configuration	3-1
Planning the SDLMUX Configuration	3-3
Planning the Physical Configuration for SDLMUX	3-3
PCI-Adapter Slot Numbers of ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 Systems	3-3
PCI-Adapter Slot Numbers of ftServer V 2404, V 4408, V 6408, and V 6512 Systems	3-5
Planning the Logical Configuration of SDLMUX-Group Interfaces	3-6

Sample SDLMUX Configuration	3-7
Software Configuration Procedures for SDLMUX	3-10
Configuring SDLMUX Devices	3-12
Creating Device Entries for SDLMUX-Group Interfaces	3-12
Field Descriptions for SDLMUX-Group Interfaces	3-12
Entries for SDLMUX-Group Interfaces	3-13
Creating Device Entries for Ports on Ethernet PCI Adapters	3-14
Field Descriptions for Ports on Ethernet PCI	
Adapters	3-14
Entries for Ports on Ethernet PCI Adapters	3-18
Creating, Installing, and Activating the <code>devices.table</code>	
File	3-20
Verifying Devices	3-20
Initializing an SDLMUX-Group Interface	3-21
Loading and Unloading SDLMUX	3-23
Automatically Loading and Unloading SDLMUX	3-23
Manually Loading and Unloading SDLMUX	3-23
Verifying SDLMUX	3-25
Defining the SDLMUX-Group Interfaces	3-25
Changing Existing Configurations	3-26
Deleting an SDLMUX-Group Interface and Its Ports	3-27
Deleting or Adding a Partner Port in an SDLMUX-Group	
Interface	3-27
Deleting a Partner Port	3-28
Adding a Partner Port	3-29
Adding a Simplexed Port	3-30
SDLMUX Requests of the <code>analyze_system</code> Subsystem	3-31

---

<b>4. Configuring STCP Devices</b>	<b>4-1</b>
Defining STCP Protocol Drivers	4-2
Defining the STCP Driver	4-2
Defining the UDP Driver	4-4
Defining the IP Driver	4-5
Defining the Loopback Driver	4-6
Defining the AF UNIX Driver	4-7
Defining STCP TELNET Devices	4-8
Defining the TLI Access-Layer Device Driver	4-9
Defining STCP TELNET Pipe Entries	4-10
Defining STCP TELNET Login Devices	4-12
Defining STCP TELNET Incoming Slave Devices	4-20
Defining STCP TELNET Outgoing Slave Devices	4-26
Field Descriptions for Clonable Outgoing Slave	
Device Entries	4-27

---

Sample Clonable Outgoing Slave Device Entry for a Printer	4-29
Sample Clonable Outgoing Slave Device Entry for a Remote Application	4-29
Creating, Installing, and Activating the <code>devices.table</code> File	4-30
Controlling Access to STCP Devices	4-30

---

<b>5. Configuring the STCP Database Files</b>	5-1
Overview of the STCP Template Database Files	5-2
The <code>bootptab</code> File and Boot-Related Information	5-4
How the <code>bootptab</code> File Is Used	5-4
How to Edit the <code>bootptab</code> File	5-5
Sample <code>bootptab</code> File	5-6
Related Information	5-7
The <code>hosts</code> File and Host Name Information	5-7
How the <code>hosts</code> File Is Used	5-7
How to Edit the <code>hosts</code> File	5-8
Sample <code>hosts</code> File	5-9
Related Information	5-10
The <code>hosts.allow</code> and <code>hosts.deny</code> Files and TCP Wrappers	5-10
How the <code>hosts.allow</code> and <code>hosts.deny</code> Files Are Used	5-11
How to Edit the <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-12
Sample <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-16
Template <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-16
Example Entries in <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-17
Related Information	5-20
The <code>inetd.conf</code> File and Information for <code>inetd</code>	5-21
How the <code>inetd.conf</code> File Is Used	5-21
How to Edit the <code>inetd.conf</code> File	5-22
Configuring the <code>tcpd</code> Daemon to Invoke the <code>swat</code> Daemon	5-24
Configuring the <code>tcpd</code> Daemon to Invoke the <code>tftpd</code> and <code>bootpd</code> Daemons	5-24
Sample <code>inetd.conf</code> File	5-24
Related Information	5-26
The <code>networks</code> File and Network Information	5-26
How the <code>networks</code> File Is Used	5-26
How to Edit the <code>networks</code> File	5-26
Sample <code>networks</code> File	5-27
The <code>nsswitch.conf</code> File and Name Server Information	5-28
How the <code>nsswitch.conf</code> File Is Used	5-28
How to Edit the <code>nsswitch.conf</code> File	5-28

Sample <code>nsswitch.conf</code> file	5-29
Related Information	5-31
The <code>ospfdconf</code> File and Routing Information	5-31
How the <code>ospfdconf</code> File Is Used	5-31
How to Edit the <code>ospfdconf</code> File	5-32
Sample <code>ospfdconf</code> Files	5-37
Related Information	5-40
The <code>protocols</code> File and Protocol Information	5-40
How the <code>protocols</code> File Is Used	5-41
How to Edit the <code>protocols</code> File	5-41
Sample <code>protocols</code> file	5-42
The <code>resolv.conf</code> File and Name Server Information	5-42
How the <code>resolv.conf</code> File Is Used	5-42
How to Edit the <code>resolv.conf</code> File	5-43
Sample <code>resolv.conf</code> file	5-46
Related Information	5-46
The <code>services</code> File and Service Information	5-46
How the <code>services</code> File Is Used	5-47
How to Edit the <code>services</code> File	5-47
Sample <code>services</code> File	5-48
The <code>snmpconf</code> File and SNMP Information	5-49
SNMP Overview	5-49
How the <code>snmpconf</code> File Is Used	5-50
How to Edit the <code>snmpconf</code> File	5-50
Sample <code>snmpconf</code> File	5-51
The <code>telnet service</code> File and TELNET Information	5-53
How the <code>telnet service</code> File Is Used	5-53
How to Edit the <code>telnet service</code> File	5-55
Sample <code>telnet service</code> File	5-56
The <code>xinetd.conf</code> File and Information for <code>xinetd</code>	5-57

---

<b>6. Starting, Verifying, and Modifying STCP</b>	6-1
Starting STCP on the Module	6-2
Modifying the <code>module_start_up.cm</code> and <code>start_stcp.cm</code> Files	6-3
Establishing the STCP Command Library Paths	6-3
Loading SDLMUX	6-3
Loading AF UNIX	6-4
Establishing the Module's STCP Host Name	6-4
Modifying and Executing the <code>start_stcp.cm</code> Command Macro	6-4
Starting STCP during the Current Bootload	6-11

---

Verifying STCP Configuration	6-12
Verifying the STREAMS Daemons	6-12
Verifying the STCP Command Library	6-13
Verifying the Host Name for the Module	6-13
Verifying the Components in the Kernel-Loadable Library	6-13
Verifying the Communications Protocols	6-14
Verifying the Drivers	6-15
Verifying the Networking Daemon Processes	6-16
Verifying the STCP Devices	6-16
Verifying the STREAMS Devices	6-17
Verifying the TELNET (Window-Terminal) Devices	6-18
Verifying Network Interfaces	6-18
Verifying Entries in the Routing Table	6-19
Verifying Basic IP Connectivity	6-19
Verifying TCP Connectivity	6-20
STCP Requests of the <code>analyze_system</code> Subsystem	6-21
Modifying STCP during Normal Operation	6-21
Adding, Deleting, or Changing the State of a Network Interface	6-21
Adding, Deleting, or Modifying a TELNET Service	6-22

---

<b>7. STCP and Security</b>	7-1
LAND Attack	7-2
Denial-of-Service Attack	7-2
Logging Discarded Packets	7-3
Port Scanning	7-4
Excerpt from <code>list_stcp_params</code> Output	7-4

---

<b>8. STCP Networking Daemons</b>	8-1
bootpd	8-4
ftpd	8-7
inetd	8-19
ospfd	8-23
rtsold	8-26
snmpd	8-29
tcpd	8-32
telnetd	8-34
tftpd	8-40
tftpd6	8-44
xinetd	8-45

<b>9. STCP Commands</b>	9-1
access_info_monitor	9-4
arp	9-10
dlmux_admin	9-15
ftp	9-25
hostname	9-35
ifconfig	9-37
IP_forwarding	9-53
ip_pair_admin	9-55
ip6addrctl	9-60
netstat	9-61
ngrep	9-106
omon	9-107
packet_monitor	9-117
ping	9-129
route	9-140
rtsol	9-154
set_udp_ordering	9-155
setkey	9-157
tcpdump	9-158
telnet	9-159
telnet_admin	9-163
tftp	9-170
tftp6	9-173
traceroute	9-174

---

<b>10. Troubleshooting Information</b>	10-1
Problem 1: Cannot Reach Hosts on the Local Subnet	10-2
Problem 2: Cannot Reach a Specific Host on the Local Subnet	10-3
Problem 3: Cannot Reach a Host on a Different Subnet	10-4
Problem 4: Outgoing TELNET or FTP Connections Fail	10-5
Problem 5: Incoming TELNET or FTP Connections Fail	10-5
Problem 6: Module Cannot Route Packets	10-6
Problem 7: No Communication Using a PCI Adapter	10-7
Problem 8: Communication Problems Associated with a Large Number of Sockets	10-7

---

<b>Appendix A. TCP/IP Networking Basics</b>	A-1
TCP/IP Networks, Subnetworks, Hosts, and Routers/Gateways	A-1
Naming Conventions	A-3



---

IP Multicast Transmission Service	A-7
Network Interfaces for STCP	A-9
Overview of Procedures for Network Interfaces	A-9
Connecting Interfaces to Multiple Networks or Subnets	A-10
Connecting Interfaces to the Same Network or Subnet	A-12
Specifying a Network Mask to Identify Subnets	A-13
Creating Variable-Length Subnets	A-14
STCP Routes	A-18
OSPF Routing	A-19
Virtual Links	A-20
Boundary Routers	A-20
NSSA	A-21
Stub Areas	A-21
Authentication	A-21
STCP Options	A-21
Obtaining IP Addresses and Domain Names	A-22
Domain Name Service	A-23

---

<b>Appendix B. Sample STCP Configuration</b>	B-1
Sample Device Entries	B-3
Sample Database Files	B-7

---

<b>Appendix C. Attaching an Application to a TELNET Slave Device</b>	C-1
Enabling Incoming (Client-Initiated) Connections	C-1
Configuring <code>telnetd</code> for an Incoming Slave Connection	C-2
Opening a TELNET Incoming Slave Device	C-4
Establishing a TELNET Session on the Client	C-5
Enabling Outgoing (Host-Initiated) Connections	C-5
Configuring Outgoing Slave Connections	C-6
Executing a Server Application on the Remote Host	C-7
Opening a TELNET Outgoing Slave Device	C-8

---

<b>Appendix D. OSPF Debug Errors</b>	D-1
--------------------------------------	-----

---

<b>Appendix E. OSPF Error Messages</b>	E-1
--	-----

---

<b>Appendix F. The MST Server</b>	F-1
Configuring TELNET Virtual Terminal Devices	F-2
Defining TELNET Virtual Terminal Devices	F-2
Fields Used in Entries for TELNET Vterm Devices	F-3
Sample Entries for TELNET Vterm Devices	F-5
Creating a <code>devices.table</code> File	F-5
Broadcasting the <code>devices.table</code> File	F-5
Issuing the <code>configure_comm_protocol</code> Command	F-6
Confirming Device Configuration	F-6
Starting the MST Server Process Automatically	F-6
The <code>telnet_msd</code> Command	F-7
<code>telnet_msd</code>	F-8

---

<b>Appendix G. IPsec Support</b>	G-1
User Documentation	G-2
Release Contents	G-2
Software Requirements	G-2
Installing the Software	G-3
Configuration Files	G-4
Using the <code>strongSwan</code> <code>man</code> Pages	G-4
Starting and Stopping the <code>strongSwan</code> Daemon	G-4
Starting the <code>strongSwan</code> Daemon During Module Startup	G-5
Starting the <code>strongSwan</code> Daemon from a Running Module	G-5
Stopping the <code>strongSwan</code> Daemon	G-5
Supported Algorithms	G-5
Supported Integrity Algorithms for AH and ESP	G-6
Supported Encryption Algorithms for ESP	G-6
Algorithm Performance	G-6
Security Policies	G-6
Default Security Policy	G-6
Ordering of Security Policies	G-9
Avoiding Problems When Tunneling	G-9
Important Considerations	G-10

---

---

# Figures

Figure 1-1.	STCP Components	1-7
Figure 3-1.	PCI-Adapter Slots on ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, V 6832 Systems	3-4
Figure 3-2.	PCI-Adapter Slots on ftServer V 2404, V 4408, V 6408, and V 6512 Systems	3-5
Figure 3-3.	Sample SDLMUX Configuration of Ethernet PCI Adapters in an ftServer V 2404, V 4408, V 6408, or V 6512 System	3-9
Figure 3-4.	SDLMUX Software Configuration Procedures	3-11
Figure 5-1.	Template <code>bootptab</code> File	5-7
Figure 5-2.	Template <code>hosts</code> File	5-10
Figure 5-3.	Template <code>hosts.allow</code> File	5-16
Figure 5-4.	Template <code>hosts.deny</code> File	5-17
Figure 5-5.	Excerpt from the Template <code>inetd.conf</code> File	5-25
Figure 5-6.	Template <code>networks</code> File	5-27
Figure 5-7.	Template <code>nsswitch.conf</code> File	5-30
Figure 5-8.	Sample <code>ospfdconf</code> File: Physical Connection to the Backbone	5-37
Figure 5-9.	System Configuration: Physical Connection to the Backbone	5-38
Figure 5-10.	Sample <code>ospfdconf</code> File: Logical Connection to the Backbone	5-38
Figure 5-11.	System Configuration: Logical Connection to the Backbone	5-40
Figure 5-12.	Template <code>protocols</code> File	5-42
Figure 5-13.	Template <code>resolv.conf</code> File	5-46
Figure 5-14.	Template <code>services</code> File	5-49
Figure 5-15.	Template <code>snmpconf</code> File	5-52
Figure 5-16.	Example <code>snmpconf</code> File with Community Strings	5-52
Figure 5-17.	Sample <code>telnetSERVICE</code> File	5-56
Figure 6-1.	Template <code>start_inetd.cm</code> File	6-10
Figure 6-2.	Template <code>start_xinetd.cm</code> File	6-10
Figure A-1.	Sample Network Configuration	A-2
Figure A-2.	Configuration with IP Addresses and Host Names	A-6
Figure A-3.	Example of Sending and Receiving an IP Multicast Packet	A-8
Figure A-4.	Connecting Network Interfaces to Different Subnets	A-11
Figure A-5.	Configuring Network Interfaces to the Same Subnet	A-13
Figure A-6.	Sample VLSN Configuration	A-16
Figure A-7.	Sample Subnet Masks	A-17

Figure A-8. Virtual Links	A-20
Figure A-9. Sample Domain Name Hierarchy	A-24
Figure B-1. Sample STCP Configuration (with Berkeley Numbers)	B-2
Figure B-2. Sample Excerpt from a <code>devices.tin</code> File	B-7
Figure B-3. Sample <code>bootptab</code> Database File	B-8
Figure B-4. Sample <code>hosts</code> Database File	B-8
Figure B-5. Sample <code>inetd.conf</code> Database File	B-9
Figure B-6. Sample <code>networks</code> Database File	B-9
Figure B-7. Sample <code>ospfdconf</code> Database File	B-10
Figure B-8. Sample <code>protocols</code> Database File	B-10
Figure B-9. Sample <code>services</code> Database File	B-11
Figure B-10. Sample <code>snmpconf</code> Database File	B-11
Figure B-11. Sample <code>telnetSERVICE</code> Database File	B-12

# Tables

Table 1-1.	How to Find STCP Information	1-4
Table 1-2.	STCP Components of the OpenVOS Kernel-Loadable Library	1-8
Table 1-3.	Components of the Loadable Driver <code>ip.cp.pm</code>	1-8
Table 1-4.	Components of the STCP Command Library	1-9
Table 1-5.	Components of the STCP Templates Subdirectory	1-13
Table 3-1.	Values for the <code>hardware_address</code> Field	3-16
Table 4-1.	Key Values for Defining Login Devices	4-15
Table 4-2.	Key Values for Defining Incoming Slave Devices	4-24
Table 5-1.	STCP Database Files	5-2
Table 5-2.	Fields in the <code>bootptab</code> File	5-6
Table 5-3.	Fields in the <code>hosts</code> File	5-9
Table 5-4.	Values for the <code>daemon</code> Field in the <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-13
Table 5-5.	Values for the <code>client</code> Field in the <code>hosts.allow</code> and <code>hosts.deny</code> Files	5-13
Table 5-6.	Fields in the <code>inetd.conf</code> File	5-23
Table 5-7.	Fields in the <code>networks</code> File	5-27
Table 5-8.	<code>database</code> Values in the <code>nsswitch.conf</code> File	5-29
Table 5-9.	<code>source</code> Values in the <code>nsswitch.conf</code> File	5-29
Table 5-10.	Entries in the <code>ospfdconf</code> File	5-32
Table 5-11.	Fields in the <code>protocols</code> File	5-41
Table 5-12.	Fields in the <code>resolv.conf</code> File	5-43
Table 5-13.	Values of the <code>options</code> Field in the <code>resolv.conf</code> File	5-45
Table 5-14.	Fields in the <code>services</code> File	5-47
Table 5-15.	System Variables in the <code>snmpconf</code> File	5-50
Table 5-16.	Fields in the <code>telnetSERVICE</code> File	5-55
Table 5-17.	Fields in a <code>SERVICES</code> File	5-57
Table 6-1.	Starting, Verifying, and Modifying STCP	6-1
Table 8-1.	STCP Daemon Processes	8-1
Table 8-2.	FTP Requests Supported by <code>ftpd</code>	8-14
Table 9-1.	STCP Commands	9-2
Table 9-2.	Values of the <code>action</code> Argument of the <code>dlmux_admin</code> Command	9-16
Table 9-3.	Common Error and Status Messages of the <code>dlmux_admin</code> Command	9-22
Table 9-4.	FTP Subcommands	9-28
Table 9-5.	Keys That Change the <code>-interval</code> Display	9-65

Table 9-6.	RFCs with <code>netstat</code> Command Output Information	9-69
Table 9-7.	OpenVOS MIB Files in <code>(master_disk)&gt;system</code> <code>&gt;vos_mibs</code>	9-70
Table 9-8.	Fields in <code>netstat</code> That Identify Connections	9-70
Table 9-9.	TCP Protocol States	9-71
Table 9-10.	Fields in the MAC Statistics Section of Ethernet Output	9-72
Table 9-11.	Fields in <code>netstat</code> That Report IF Statistics	9-74
Table 9-12.	Fields in <code>netstat</code> That Report ICMP Statistics with Overall IPv4 and IPv6 Counts	9-75
Table 9-13.	Examples of ICMP Message Types Displayed with Incoming and Outgoing Packet Statistics	9-76
Table 9-14.	Fields Displayed by <code>netstat -protocol tcp/ext</code>	9-77
Table 9-15.	Fields in <code>netstat</code> That Report UDP Statistics	9-92
Table 9-16.	Fields in <code>netstat</code> That Report AF UNIX Statistics	9-93
Table 9-17.	Problems Indicated by Fields in the <code>netstat</code> Statistics	9-95
Table 9-18.	Fields in a Destination File	9-108
Table 9-19.	The <code>omon</code> Subcommands: Configuring the Session	9-109
Table 9-20.	The <code>omon</code> Subcommands: Monitoring Modules and Routers	9-110
Table 9-21.	Values for the <code>flags</code> Subcommand	9-113
Table 9-22.	TELNET Subcommands	9-161
Table 9-23.	<code>telnet</code> service File Fields and Corresponding <code>telnet_admin</code> Arguments	9-167
Table 9-24.	Message Characters in <code>traceroute</code> Command Output for IPv6 Addresses	9-178
Table A-1.	STCP Options	A-21
Table D-1.	Errors Displayed by the <code>list_errors</code> Subcommand	D-1
Table F-1.	Severity Levels of MST Status Messages	F-10
Table G-1.	Packet Types Needed for Internet Communication	G-7

---

## Preface

The *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419) documents the tasks required to configure and manage STREAMS TCP/IP (STCP).

This manual is intended for network administrators. It may also be useful to systems programmers who are designing distributed applications or networking software with specific configuration and/or administration requirements.

For additional STCP information, see the following manuals.

- *OpenVOS STREAMS TCP/IP Migration Guide* (R418)
- *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420)
- *OpenVOS STREAMS TCP/IP User's Guide* (R421)
- *OpenVOS POSIX.1 Reference Guide* (R502)

### Manual Version

This manual is a revision. Change bars, which appear in the margin, note the specific changes to text since the previous publication of this manual.

This manual includes the following changes:

- Support for OpenVOS Release 19.3.1
- Support for ftServer V 2810 and V 4820 systems. See “[PCI-Adapter Slot Numbers of ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 Systems](#)” on page 3-3.
- Changes to the [arp](#), [route](#), [tcpdump](#), and [telnet\\_msd](#) commands
- [Appendix G, “IPsec Support,”](#) is new

#### NOTE

Contact your account representative for information about the availability of ftServer V 6728 systems on OpenVOS Release 19.2.x and later.

## Manual Organization

This manual contains the following chapters and appendixes.

[Chapter 1](#) provides an overview of STCP. It describes the components of the STCP product and outlines the basic hardware and software requirements.

[Chapter 2](#) summarizes the STCP configuration procedures.

[Chapter 3](#) describes how to configure SDLMUX.

[Chapter 4](#) describes the configuration requirements for the different types of STCP devices.

[Chapter 5](#) describes how to specify information in the STCP database files.

[Chapter 6](#) describes how to start, modify, and verify STCP on the module.

[Chapter 7](#) discusses STCP and security.

[Chapter 8](#) describes the STCP networking daemons.

[Chapter 9](#) describes the STCP commands.

[Chapter 10](#) describes how to identify and fix some common STCP configuration errors.

[Appendix A](#) provides an overview of TCP/IP networking basics, such as IP addresses, subnet masks, and routing.

[Appendix B](#) provides a sample STCP configuration.

[Appendix C](#) explains how applications must attach to STCP TELNET slave devices.

[Appendix D](#) describes the messages returned by the `omon` subcommand, `list_errors`.

[Appendix E](#) describes the OSPF error messages that appear in the `(master_disk)>system>step>logs>ospfd.io` file.

[Appendix F](#) describes the Multisession TELNET (MST) server, `telnet_msd` (use this server only with legacy applications).

[Appendix G](#) describes the strongSwan for OpenVOS implementation of IPsec.



## Related Manuals

See the following manuals for information about ftServer hardware.

- the site planning guide for your system:
  - *Stratus ftServer V 2608, V 4612, and V 6616 Systems: Site Planning Guide* (R772)
  - *Stratus ftServer V 6624 System: Site Planning Guide* (R763)
  - *Stratus ftServer V 2404, V 4408, and V 6408 Systems: Site Planning Guide* (R645)
  - *Stratus ftServer V 6512 System: Site Planning Guide* (R675)
  - *Stratus ftServer V 2810, V 4820, and V 6832 Systems: Site Planning Guide* (R794)
- the operation and maintenance guide for your system:
  - *Stratus ftServer V 2608, V 4612, and V 6616 Systems: Operation and Maintenance* (R773)
  - *Stratus ftServer V 6624 System: Operation and Maintenance* (R764)
  - *Stratus ftServer V 2404, V 4408, and V 6408 Systems: Operation and Maintenance Guide* (R646)
  - *Stratus ftServer V 6512 System: Operation and Maintenance Guide* (R674)
  - *Stratus ftServer V 2810, V 4820, and V 6832 Systems: Operation and Maintenance* (R795)
- See *Stratus ftServer V Series Systems: V 6728 Hardware Supplement* (R790) for information about ftServer V 6728 systems.
- See the *Stratus ftServer: Network I/O Enclosure Guide* (R608) for information about the network I/O enclosure.

### NOTE

---

The network I/O enclosure is not available for purchase with V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, or V 6832 systems. This release supports the network I/O enclosures that are already installed on V 2404, V 4408, V 6408, and V 6512 systems. If you are interested in migrating an existing network I/O enclosure to a V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, or V 6832 system, contact your account representative.

- See the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) for information about ftServer systems and their PCI adapters.
- See the *Stratus ftServer Systems: Peripherals Site Planning Guide* (R582) for information about peripherals, including disk and tape drives.

See the following Stratus manuals for information about basic OpenVOS administrative procedures.

- *OpenVOS System Administration: Administering and Customizing a System* (R281)
- *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282)
- *OpenVOS System Administration: Registration and Security* (R283)
- *OpenVOS System Administration: Disk and Tape Administration* (R284)
- *OpenVOS System Administration: Backing Up and Restoring Data* (R285)
- *VOS System Administration: Administering the Spooler Facility* (R286)
- *OpenVOS System Administration: Configuring a System* (R287)

See the following Stratus manuals for additional OpenVOS information.

- *OpenVOS Commands Reference Manual* (R098)
- *OpenVOS Installation Guide* (R386)

See also the following third-party documentation.

- *TCP/IP Illustrated*. Vols 1 and 2. Gary Wright and W. Richard Stevens (Reading, MA: Addison-Wesley Publishing Company, 1995)
- Request for Comments (RFCs), available at <http://www.ietf.org>:
  - RFC-1253 and RFC-2328 for Open Shortest Path First (OSPF)
  - RFC-2401 for Internet Protocol Security (IPsec)
  - RFC-4291 for IPv6 addressing architecture

## Notation Conventions

This manual uses the following notation conventions.

### Warnings, Cautions, and Notes

Warnings, cautions, and notes provide special information and have the following meanings:



#### **WARNING** \_\_\_\_\_

A warning indicates a hazardous situation that, if not avoided, could result in death or serious injury.



#### **AVERTISSEMENT** \_\_\_\_\_

Un avertissement indique une situation dangereuse qui, si pas évitée, pourrait entraîner la mort ou des blessures graves.



#### **CAUTION** \_\_\_\_\_

A caution indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.



#### **MISE EN GARDE** \_\_\_\_\_

Une mise en garde indique une situation dangereuse qui, si pas évitée, pourrait entraîner des blessures mineures ou modérées.

#### **NOTICE** \_\_\_\_\_

A notice indicates information that, if not acted on, could result in damage to a system, hardware device, program, or data, but does not present a health or safety hazard.

#### **NOTE** \_\_\_\_\_

A note provides important information about the operation of an ftServer system or related equipment or software.

## Typographical Conventions

The following typographical conventions are used in this manual:

- Italics introduces or defines new terms. For example:

The *master disk* is the name of the member disk from which the module was booted.

- Boldface emphasizes words in text. For example:

Every module **must** have a copy of the `module_start_up.cm` file.

- Monospace represents text that would appear on your terminal's screen (such as commands, subroutines, code fragments, and names of files and directories). For example:

```
change_current_dir (master_disk)>system>doc
```

- Monospace italic represents terms that are to be replaced by literal values. In the following example, the user must replace the monospace-italic term with a literal value.

```
list_users -module module_name
```

- Monospace bold represents user input in examples and figures that contain both user input and system output (which appears in monospace). For example:

```
display_access_list system_default
```

```
%dev#m1>system>acl>system_default
```

```
w  *.*
```

## Format for Commands and Requests

Stratus manuals use the following format conventions for documenting commands and requests. (A *request* is typically a command used within a subsystem, such as `analyze_system`.) Note that the command and request descriptions do not necessarily include each of the following sections.

**A** — `add_disk`

**B** — **Privileged**

**C** — **Purpose**

The `add_disk` command tells the operating system on the current module to recognize the specified logical volume for the duration of the current bootload.

**D** — **Display Form**

```
----- add_disk -----
disk_name: 
module_name:  current_module
```

**E** — **Command Line Form**

```
add_disk disk_name
[ module_name ]
```

**F** — **Arguments**

► `disk_name`

**G** — **Required**

The name of the logical volume to be recognized for the current bootload.

**H** — .  
.  
.

**A** **name**

The name of the command or request is at the top of the first page of the description.

**B** **Privileged**

This notation appears after the name of a command or request that can be issued only from a privileged process.

**C** **Purpose**




Explains briefly what the command or request does.

**D** **Display Form**

Shows the form that is displayed when you type the command or request name followed by `-form` or when you press the key that performs the `DISPLAY FORM` function. Each field in the form represents a command or request argument. If an argument has a default value, that value is displayed in the form.

The following table explains the notation used in display forms.

**The Notation Used in Display Forms**

Notation	Meaning
	Required field with no default value.
	The cursor, which indicates the current position on the screen. For example, the cursor may be positioned on the first character of a value, as in  ll.
<i>current_user</i> <i>current_module</i> <i>current_system</i> <i>current_disk</i>	The default value is the current user, module, system, or disk. The actual name is displayed in the display form of the command or request.

**E Command-Line Form**

Shows the syntax of the command or request with its arguments. You can display an online version of the command-line form of a command or request by typing the command or request name followed by `-usage`.

The following table explains the notation used in command-line forms. In the table, the term *multiple values* refers to explicitly stated separate values, such as two or more object names. Specifying multiple values is **not** the same as specifying a star name. When you specify multiple values, you must separate each value with a space.

## The Notation Used in Command-Line Forms

Notation	Meaning
<code>argument_1</code>	Required argument.
<code>argument_1...</code>	Required argument for which you can specify multiple values.
$\left\{ \begin{array}{l} \text{argument}_1 \\ \text{argument}_2 \end{array} \right\}$	Set of arguments that are mutually exclusive; you must specify one of these arguments.
<code>[argument_1]</code>	Optional argument.
<code>[argument_1]...</code>	Optional argument for which you can specify multiple values.
$\left[ \begin{array}{l} \text{argument}_1 \\ \text{argument}_2 \end{array} \right]$	Set of optional arguments that are mutually exclusive; you can specify only one of these arguments.
<b>Note:</b> Dots, brackets, and braces are not literal characters; you should <b>not</b> type them. Any list or set of arguments can contain more than two elements. Brackets and braces are sometimes nested.	

### F Arguments

Describes the command or request arguments. The following table explains the notation used in argument descriptions.

### G The Notation Used in Argument Descriptions

Notation	Meaning
<code>CYCLE</code>	This argument has predefined values. In the display form, you display these values in sequence by pressing the key that performs the <code>CYCLE</code> function.
<b>Required</b>	<p>You cannot issue the command or request without specifying a value for this argument.</p> <p>If an argument is required but has a default value, it is not labeled <b>Required</b> since you do not need to specify it in the command-line form. However, in the display form, a required field must have a value—either the displayed default value or a value that you specify.</p>
<b>(Privileged)</b>	Only a privileged process can specify a value for this argument.

- H** The following additional headings may appear in the command or request description: Explanation, Error Messages, Examples, and Related Information.

**Explanation**

Explains how to use the command or request and provides supplementary information.

**Error Messages**

Lists common error messages with a short explanation.

**Examples**

Illustrates uses of the command or request.

**Related Information**

Refers you to related information (in this manual or other manuals), including descriptions of commands, subroutines, and requests that you can use with or in place of this command or request.

## Getting Help

If you have a technical question about ftServer system hardware or software, try these online resources first:

- **Online documentation at the StrataDOC Web site.** Stratus provides complimentary access to StrataDOC, an online-documentation service that enables you to view, search, download, and print customer documentation. You can access StrataDOC at the following Web site:

<http://stratadoc.stratus.com>

- **Online support from Stratus Customer Service.** You can find the latest technical information about an ftServer system through online product support at the Customer Support Web site:

<http://www.stratus.com/go/support>

The Service Portal provides access to Knowledge Base articles for all Stratus product lines. You can locate articles by performing a simple or advanced keyword search, viewing recent articles or top FAQs, or browsing a product and category.

To log in to the Service Portal, enter your employee user name and password or, if you have not been provided with a login account, click **Register Account**. When registering a new account, ensure that you specify an email address from a company that has a service agreement with Stratus.

If you cannot resolve your questions with these online self-help resources, and the ftServer system is covered by a service agreement, contact the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative. To contact



the CAC, use the Service Portal to log a support request. Click **Customer Support** and **Add Issue**, and then complete the **Create Issue** form. A member of our Customer Service team will be glad to assist you.

## Commenting on This Manual

You can comment on this manual using one of the following methods. When you submit a comment, be sure to provide the manual's name and part number, a description of the problem, and the location in the manual where the affected text appears.

- From StrataDOC, click the **site feedback** link at the bottom of any page. In the pop-up window, answer the questions and click **Submit**.
- From any email client, send email to `comments@stratus.com`.
- From the Stratus Customer Service Portal, log on to your account and create a new issue.

Stratus welcomes any corrections and suggestions for improving this manual.



---

# Chapter 1

## Overview of STREAMS TCP/IP (STCP)

STREAMS TCP/IP (STCP) is a STREAMS-based implementation of the standard Transmission Control Protocol/Internet Protocol (TCP/IP) family of protocols for Stratus modules. The STCP product functions within a standard STREAMS environment. It relies on OpenVOS communications drivers, device drivers, and Stratus communications hardware to support the Ethernet local area network (LAN) architecture.

This manual describes how to configure and manage STCP on a Stratus module running OpenVOS. This chapter, which contains the following sections, provides an overview of STCP.

- [“Key Features” on page 1-1](#)
- [“How to Find STCP-Specific Information” on page 1-4](#)
- [“Components of STCP” on page 1-5](#)
- [“Related Drivers” on page 1-15](#)
- [“Hardware Requirements” on page 1-17](#)
- [“Software Requirements and Considerations” on page 1-17](#)

### Key Features

The following list identifies key features of STCP.

- STCP provides various security features.
- OpenVOS requires STCP for connections to the network I/O enclosure and to the maintenance network, which supports uninterruptible power supply (UPS) units, the Remote Service Network (RSN), storage devices, and a Fibre Channel (FC) switch.
- STCP includes TCP wrappers functionality, which provides authentication and logging capabilities that enable STCP to control client access to TELNET and the File Transfer Protocol (FTP), and to any services that the `inetd` daemon starts (for example, BOOTP and TFTP). STCP includes two versions of an application programming interface (API) for TCP wrappers functionality: an IPv6-enabled

version and a legacy version. For information, see “[Checklist for TCP Wrappers Functionality](#)” on page 2-7.

For TCP wrappers functionality, STCP also provides the `tcpd.h` header file in the `(master_disk)>system>include_library` directory.

- The STCP `inetd` daemon complies with Part 1 of the IEEE POSIX® standard (commonly known as the POSIX.1 standard). The `inetd` daemon supports only the IPv4 protocol.

STCP provides the `xinetd` daemon for IPv6 protocol support.

- STCP enables the module to act as a router if you have connections to multiple networks or subnetworks. For IPv4, this feature is enabled by default, as described in “[Software Requirements and Considerations](#)” on page 1-17.
- STCP provides a standard socket programming interface.
- STCP consists of one product that includes the protocol stack, the utilities, and the programming interface.
- STCP is required for some products such as OpenVOS Open StrataLINK.
- STCP reduces the number of configuration procedures by relying on STREAMS to load the components in the protocol stack and any related STREAMS drivers.
- STCP supports the IP multicast transmission service (level 2).
- STCP supports variable-length subnets (VLSN) to provide flexibility with the configuration of subnets and subnet masks.
- STCP supports dynamic routing with the Open Shortest Path First daemon (OSPFD or `ospfd`), a network daemon that implements the OSPF protocol. The `ospfd` daemon supports only the IPv4 protocol.
- STCP supports a wide range of standard TCP/IP applications, such as TELNET and FTP, and has the platform necessary to support additional standard applications.

## NOTES

---

1. STCP supports two implementations of a TELNET server: the standard STCP TELNET server (started by the `telnetd` command) and the Multisession TELNET (MST) server (started by the `telnet_msd` command). The MST server is for legacy applications only; for information, see [Appendix F, “The MST Server.”](#) All references to TELNET in this manual are to the standard STCP TELNET server, unless otherwise noted. The TELNET protocol is non-secure; use the secure shell (SSH) utility for greater security (see Note 2).

2. The standard STCP TELNET server does not provide IPv6 protocol support; instead, it supports only IPv4. To support incoming login requests and incoming slave requests with the IPv6 protocol, use the SSH utility. For information on SSH, see the *Software Release Bulletin: Internet Security Pack for OpenVOS, Release 4.0.2* (R660).
  3. The STCP TELNET client (started by the `telnet` command) provides support for both the IPv4 and IPv6 protocols, though the TELNET protocol is non-secure.
- STCP offers the Simple Network Management Protocol (SNMP) daemon, whose information can be accessed by remote network management stations.
  - STCP offers a complete STREAMS-based protocol stack with a single kernel-loadable driver.
  - STCP supports window scaling, Round-Trip Time Measurement (RTTM), and Protect Against Wrapped Sequence Numbers (PAWS). For information about this support, see the *OpenVOS System Analysis Manual* (R073) for information about the `tcp_rfc_1323_policy`, `tcp_window_scale`, and `tcp_SACK_policy` parameters in the description of the `list_stcp_params` request of `analyze_system`.
  - STCP supports both the IPv4 and IPv6 protocols, functionality that is sometimes referred to as *dual stack*. An IPv6 address is 16 bytes long and written in sequences of two-byte hexadecimal numbers separated by colons. An IPv4 address is 4 bytes long and typically written as four decimal values separated by dots. Individual commands and daemon processes may provide support for only IPv4, only IPv6, or both IPv4 and IPv6 protocols, as noted in the descriptions of the individual commands and daemon processes.

For information on configuring IPv6 support, see “[Checklist for Enabling IPv6 Support](#)” on page 2-9.

Support for IPv6 addresses is available to POSIX applications (see *OpenVOS POSIX.1 Reference Guide* (R502)).

- STCP supports the Internet Key Exchange (IKE) implementation from strongSwan. For information, see <http://www.strongswan.org/>. Also see [Appendix G](#) for more information about strongSwan.
- STCP supports path MTU discovery. The *maximum transmission unit* (MTU) is the packet size. Path MTU discovery is an algorithm for determining the largest usable MTU between two hosts that are not on a local subnet. The minimum MTU for IPv6 is 1280 and for IPv4 is 576 (though tunable). You can adjust MTU using commands

such as `route` and `ip_pair_admin`. The `ip_pair_admin` command also enables you to control path MTU discovery.

## How to Find STCP-Specific Information

[Table 1-1](#) identifies topics related to configuring and managing STCP on a Stratus module and explains where to find the information.

**Table 1-1. How to Find STCP Information** (Page 1 of 2)

For Information about...	See...
Configuration summary (procedural)	<a href="#">Chapter 2</a>
Commands (for applications such as <code>netstat</code> )	<a href="#">Chapter 9</a> <i>OpenVOS STREAMS TCP/IP User's Guide</i> (R421), which describes applications such as FTP and TELNET
Communications drivers and device drivers	<a href="#">“Related Drivers” on page 1-15</a>
Daemons (server processes such as <code>telnetd</code> and <code>inetd</code> )	<a href="#">Chapter 8</a>
Database files	<a href="#">Chapter 5</a>
Device configuration (for STCP devices)	<a href="#">Chapter 4</a>
Hardware requirements	<a href="#">“Hardware Requirements” on page 1-17</a>
IP multicast	<a href="#">“IP Multicast Transmission Service” on page A-7</a>
Migrating from OS TCP/IP to STCP	<i>OpenVOS STREAMS TCP/IP Migration Guide</i> (R418), which describes the differences between STCP and OS TCP/IP, including how to modify existing OS TCP/IP applications to run with STCP, differences in user applications, and differences in the configuration procedures
PCI adapters	<i>Stratus ftServer V Series Systems: PCI Adapter Guide</i> (R679)

**Table 1-1. How to Find STCP Information** (Page 2 of 2)

For Information about...	See...
Programming information (socket library)	<p><i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420), which describes the socket programming interface that enables programmers to write STCP-based applications</p> <p>POSIX documentation at <a href="http://www.opengroup.org">www.opengroup.org</a></p> <p>RFC-3493 (POSIX standard)  RFC-3542 (Advanced Sockets API for IPv6)  RFC-3678 (Socket Interface Extensions for Multicast Source Filters)</p>
RFCs	<p>To view an RFC, perform the following steps:</p> <ol style="list-style-type: none"> <li>1. In a Web browser, open <a href="http://rfc-editor.org">http://rfc-editor.org</a>.</li> <li>2. Click on <b>Search for an RFC and its meta-data</b>.</li> <li>3. Enter the RFC number in the search field, select <b>All PDF</b>, and then press the SEARCH button. The display presents a table with a link to the RFC.</li> </ol>
Sample STCP configuration	<a href="#">Appendix B</a>
Security	<a href="#">Chapter 7</a>
Software components of STCP	<a href="#">“Components of STCP” on page 1-5</a>
Software components that work with STCP (related drivers)	<a href="#">“Related Drivers” on page 1-15</a>
Software requirements	<a href="#">“Software Requirements and Considerations” on page 1-17</a>
Starting, verifying, and modifying STCP	<a href="#">Chapter 6</a>
TCP/IP networking basics	<a href="#">Appendix A</a>
Troubleshooting information	<a href="#">Chapter 10</a>

## Components of STCP

STCP consists of the following main software components and related features.

- The STCP kernel-loadable components are the protocol drivers that comprise the STCP protocol stack as well as the communications drivers and lower-level device drivers that communicate between the protocol stack and the network interface cards (NICs). These components reside in the general OpenVOS directory (master\_disk)>system>kernel\_loadable\_library.
- The STCP startup command macro enables you to start the STCP processes during the current bootload. You should use the startup macro to start STCP at

each bootload from the `module_start_up.cm` file. A template for this command macro resides in the directory `(master_disk)>system>stcp>templates`. Use this template to create your own module-specific versions of the macros. Be sure to place the final versions of the macros in the directory `(master_disk)>system>stcp`.

- The `start_inetd.cm` and `stop_inetd.cm` command macros enable you to start and stop the `inetd` and `xinetd` daemon processes. You must be logged in as the user `root` in order to issue the `start_inetd.cm` command macro. You must be logged in as privileged in order to issue the `stop_inetd.cm` command macro.
- The STCP command library consists of the applications and networking daemons available with STCP. This command library has the path name `(master_disk)>system>stcp>command_library`.
- The STCP template database files show the format of the files that define various aspects of STCP on the module. They help you create your own files that define your specific network configuration. The template files reside in the directory `(master_disk)>system>stcp>templates`. Do not modify the template files; you must create your own versions of the files and place them in the directory `(master_disk)>system>stcp`.
- The STCP log files associated with the individual daemon processes (and the internal `start_stcp_stack` command issued as part of the `start_stcp.cm` command macro) are placed in the `(master_disk)>system>stcp>logs` directory, which is initially empty.
- The STCP application programming interface (API) consists of a socket library that enables programmers to write C-language STCP applications that use the standard TCP/IP protocols to communicate with other hosts in a TCP/IP network. The STCP socket library is ANSI C- and POSIX.1-compliant. Stratus strongly recommends that you compile your applications with the OpenVOS Standard C compiler (invoked with the `cc` command), which is ANSI C-compliant. The include files (header files) that STCP uses reside in subdirectories of the directory `(master_disk)>system>include_library`. (The directory contains a number of include-file subdirectories to help you build your own application using the STCP header files.) For complete information on the STCP socket functions, see the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420).

The `(master_disk)>system>stcp_object_library` includes a `tcpwrappers` subdirectory, which contains object modules for use by legacy open-source applications that embed TCP wrappers functionality. The header file (`tcpd.h`) that such applications require is located in the `(master_disk)>system>include_library` directory.

The TCP wrappers object modules for POSIX applications are in the POSIX object library. For information, see the *OpenVOS POSIX.1 Reference Guide* (R502).



- STCP provides the directory (master\_disk)>system>stcp>tftp\_default for files transferred by a TFTP user who does not specify a target directory.
- The number of TCP sockets available with STCP is 32,000 (or more, based on available memory).

Figure 1-1 shows several components of STCP.

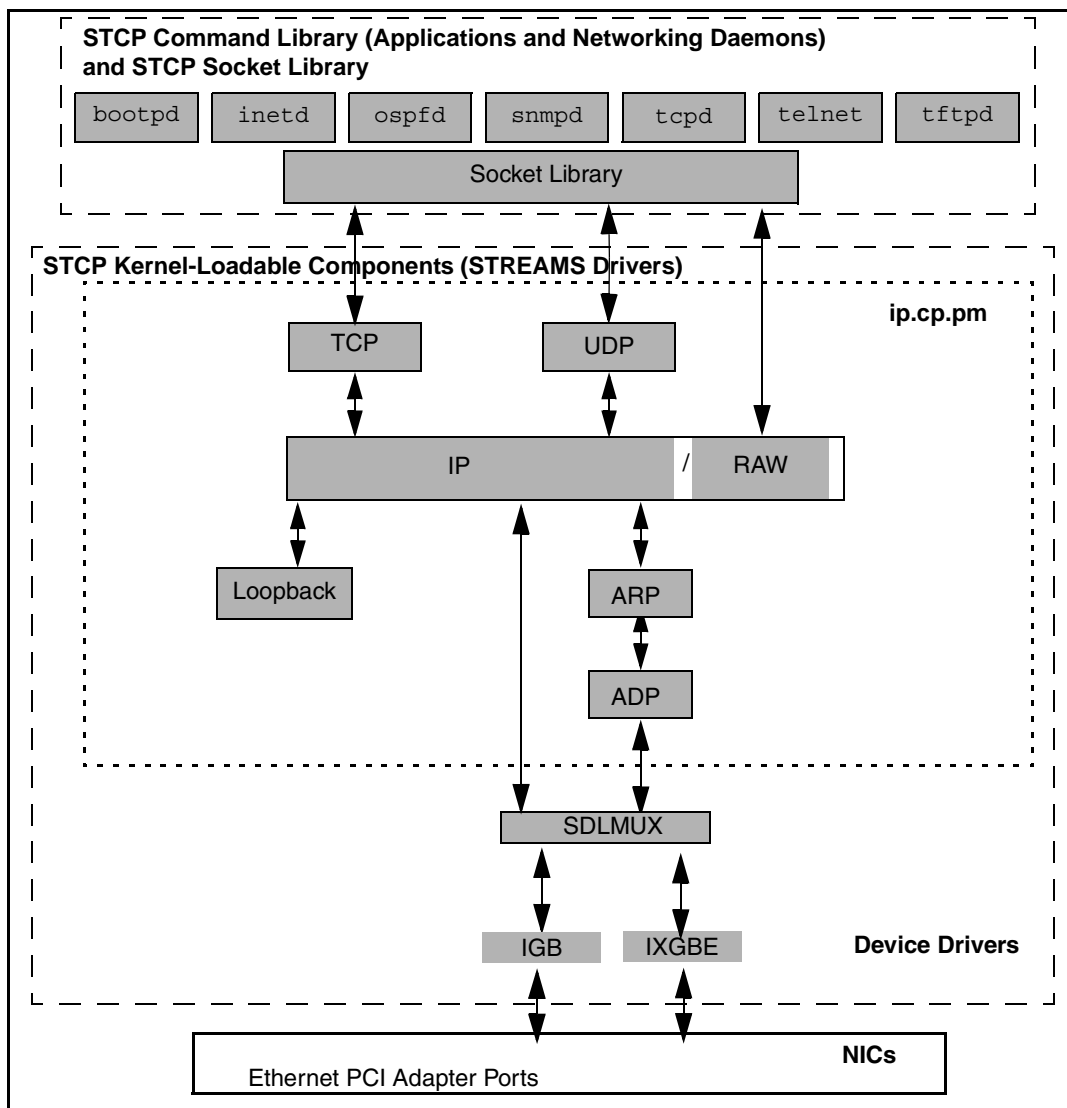


Figure 1-1. STCP Components

[Table 1-2](#) lists the components that reside in the OpenVOS kernel-loadable library ((master\_disk)>system>kernel\_loadable\_library).

**Table 1-2. STCP Components of the OpenVOS Kernel-Loadable Library**

Component	Description
af_unix.cp.pm	This AF UNIX (af_unix) driver is required for POSIX-compliant UNIX®-domain sockets.
igb.cp.pm	IGB is a lower-level device driver that functions below SDLMUX to support certain Ethernet PCI-Express adapters.
ip.cp.pm	IP (which supports IPv6) contains components listed in <a href="#">Table 1-3</a> . Links for adp, loop, stcp, and udp exist to enable legacy configuration commands.
ixgbe.cp.pm	IXGBE is a lower-level device driver that functions below SDLMUX to support certain Ethernet PCI-Express adapters.
tnmod.cp.pm	This STREAMS driver implements the TELNET protocol and handles option negotiation. It works with the <a href="#">telnetd</a> daemon process.
tpipe.cp.pm	This TELNET pipe driver passes information between the <a href="#">telnetd</a> daemon process and the <a href="#">telnet_admin</a> command to manage local TELNET services.

[Table 1-3](#) lists the components that of the loadable driver ip.cp.pm (which supports IPv6).

**Table 1-3. Components of the Loadable Driver ip.cp.pm (Page 1 of 2)**

Component	Description
adp	This STREAMS driver converts the packets that it receives from the Address Resolution Protocol (ARP) module to Data-Link Provider Interface (DLPI) format, Version 1.2 or 2.0.
arp	This STREAMS driver implements the ARP protocol, which resolves the destination hardware address for outgoing Internet packets by converting an Internet address to the corresponding MAC address. (The MAC address is a physical address unique to the NIC that connects the module to the LAN.) It also responds to incoming ARP request packets and allows users to access the ARP tables. It conforms to RFC 1042.

**Table 1-3. Components of the Loadable Driver `ip.cp.pm`** (Page 2 of 2)

Component	Description
<code>ip</code>	This STREAMS driver implements the Internet Protocol (IP) to provide network-layer routing, packet delivery, packet fragmentation, and packet reassembly. It complies with RFC 791 and follow-on specifications (except for the timestamp option). It implements the Internet Control Message Protocol (ICMP) and the Internet Group Message Protocol (IGMP). For IPv6, this layer implements the Internet Control Message Protocol6 (ICMP6), Neighbor Discovery Protocol (NDP) and Multi-cast Listener Discovery (MLD). Raw I/O is also handled within the IP module.
<code>loop</code>	This STREAMS driver is a loopback driver that receives datagrams from the IP driver and uses an Internet loopback address of 127.0.0.1 to send the datagrams back upstream. It has only one instantiation for the STCP stack.
<code>sdlmux</code>	The STREAMS Data-Link Multiplexer (SDLMUX) is an OpenVOS communications driver that provides failover support for STCP, the device driver (IGB or IXGBE), and the communications hardware (for example, Ethernet PCI adapters).
<code>stcp</code>	This STREAMS driver implements the Transport Control Protocol (TCP) and provides the interface between the socket library and the IP driver. It supports the implementation of TCP outlined in RFC 793 and follow-on specifications.
<code>udp</code>	This STREAMS driver implements the User Datagram Protocol (UDP) to provide datagram services. It complies with the UDP implementation outlined in RFC 768 and follow-on specifications. The <code>set_udp_ordering</code> command specifies the UDP message ordering policy.

Table 1-4 lists the components that reside in the STCP command library ((`master_disk`)>`system`>`stcp`>`command_library`).

**Table 1-4. Components of the STCP Command Library** (Page 1 of 4)

Component	Description
<code>arp.pm</code>	This program module implements the <code>arp</code> command, which displays or sets an entry in the address-translation table used by the ARP module.
<code>bootpd.pm</code>	This program module implements the <code>bootpd</code> daemon process, which implements the BOOTP protocol and services boot-file requests. The <code>inetd</code> daemon handles the initial client connection for the BOOTP service, but then the <code>inetd</code> daemon executes the <code>bootpd.pm</code> program module and runs <code>bootpd</code> as a separate daemon, under the user <code>nobody.nobody</code> .
<code>ftp.pm</code>	This program module implements the <code>ftp</code> command, which implements FTP to perform file-transfer operations with remote hosts.

**Table 1-4. Components of the STCP Command Library** (Page 2 of 4)

Component	Description
<code>ftpd.pm</code>	This program module implements the <code>ftpd</code> daemon process, which handles incoming FTP requests.
<code>ftpd_ch.pm</code>	This program module creates an FTP child process for each FTP client session managed by the <code>ftpd</code> daemon process.
<code>hostname.pm</code>	This program module implements the <code>hostname</code> command, which sets and/or displays the name of the current host module. After <code>hostname</code> is issued, it generates a file called <code>host</code> in the <code>(master_disk)&gt;system&gt;stcp</code> directory. To set the host name of the module, issue the <code>hostname</code> command from the <code>module_start_up.cm</code> file.
<code>ifconfig.pm</code>	This program module implements the <code>ifconfig</code> command, which enables you to add, delete, verify, or change the state of a NIC. To permanently establish each logical interface running STCP, issue an <code>ifconfig</code> command from your <code>start_stcp.cm</code> command macro.
<code>inetd.pm</code>	<p>This program module implements the <code>inetd</code> daemon process, a super server that listens on many ports. This daemon process is deprecated because it only supports IPv4. Use <code>xinetd</code> for IPv6 support.</p> <p>When <code>inetd</code> receives a connection request, it determines the target service and invokes the service to handle the request. The <code>inetd</code> daemon process must run under the user <code>root</code>. (Beginning in VOS Release 14.5, the <code>inetd</code> daemon process is POSIX.1 compliant; in previous releases of VOS, the <code>inetd</code> daemon process was not POSIX.1 compliant.)</p>
<code>IP_forwarding.pm</code>	This program module implements the <code>IP_forwarding</code> command, which enables you to verify or change the variable that determines whether the module can act as an IPv4 router or simply as a host. By default, a module configured for STCP can act as a router. Use the <code>IP_forwarding</code> command to turn off forwarding.
<code>ip_pair_admin.pm</code>	This program module implements the <code>ip_pair_admin</code> command, which enables you to control path MTU discovery on a network path between two IP addresses.
<code>ip6addrctl.pm</code>	This program module implements the <code>ip6addrctl</code> command, which configures the IPv4 and IPv6 address selection policy.
<code>netstat.pm</code>	This program module implements the <code>netstat</code> command, which displays status information about the protocols, connections, routes, and configured NICs.
<code>ngrep.pm</code>	This program module implements the <code>ngrep</code> command, which is a network packet analyzer.

**Table 1-4. Components of the STCP Command Library** (Page 3 of 4)

Component	Description
<code>omon.pm</code>	This program module implements the <code>omon</code> command, which starts an Open Shortest Path First (OSPF) monitoring session that allows you to monitor OSPF on this and other routers.
<code>ospfd.pm</code>	This program module implements the <code>ospfd</code> daemon process, an implementation of the OSPF routing protocol.
<code>packet_monitor.pm</code>	This program module implements the <code>packet_monitor</code> command, which monitors traffic in the network stack and displays information about the packets handled by a NIC. Since this command provides only IPv4 protocol support, use <code>tcpdump</code> instead.
<code>ping.pm</code>	This program module implements the <code>ping</code> command, which sends an ICMP echo packet (for IPv4) or an ICMPv6 packet (for IPv6) to verify that a specified host is active.
<code>route.pm</code>	This program module implements the <code>route</code> command, which enables you to set and display routes.
<code>rtsold.pm</code>	This program module implements the <code>rtsold</code> daemon process, which sends ICMPv6 Router Solicitation messages on specified interfaces. Use the <code>rtsold</code> daemon process for all IPv6 interfaces that are not configured in a router.
<code>snmpd.pm</code>	This program module implements the <code>snmpd</code> networking daemon process, which supports the Simple Network Management Protocol (SNMP). It supports SNMP managers and collects data from the protocol stack.
<code>start_stcp_stack.pm</code>	This program module implements an internal command issued from the <code>start_stcp.cm</code> command macro to establish the STCP protocol stack. It must not be used in any other way.
<code>sync_cfgd.pm</code>	This program module implements the <code>sync_cfgd</code> daemon process, which is a Dynamic Host Configuration Protocol (DHCP) server for network I/O enclosure devices. The <code>sync_cfgd</code> daemon automatically assigns IP addresses to these devices based on the information in the <code>sync_cfgd.conf</code> file. For complete information, see the <i>Stratus ftServer: Network I/O Enclosure Guide</i> (R608).
<code>tcpd.pm</code>	This program module implements the <code>tcpd</code> daemon process, which authenticates clients that request services specified in the <code>inetd.conf</code> file. (Client authentication is part of TCP wrappers functionality.)
<code>tcpdump.pm</code>	This program module implements the <code>tcpdump</code> command, which provides an industry-standard command-line packet analyzer.

**Table 1-4. Components of the STCP Command Library** (Page 4 of 4)

Component	Description
<code>telnet.pm</code>	<p>This program module implements the <code>telnet</code> command, which implements the TELNET client. It gives STCP users a virtual terminal connection to a remote system, and it supports both IPv4 and IPv6 protocols.</p> <p>The <code>telnet</code> protocol is insecure. For more secure communications, use SSH.</p>
<code>telnet_admin.pm</code>	This program module implements the <code>telnet_admin</code> command, which enables you to add, delete, list, or modify a TELNET service while the <code>telnetd</code> daemon process is running. This command automatically and permanently updates the <code>telnet</code> service and <code>services</code> database files. (You must create the appropriate device entries for the service before you issue the command.)
<code>telnet_msd.pm</code>	This program module implements the <code>telnet_msd</code> daemon process, which starts the MST daemon (for use with legacy applications only).
<code>telnetd.pm</code>	This program module implements the <code>telnetd</code> daemon process, which handles login, incoming slave, and outgoing slave connections for the standard STCP implementation of TELNET. It supports only the IPv4 protocol; it does not support IPv6.
<code>tftp.pm</code>	This program module implements the <code>tftp</code> command, which implements the TFTP protocol and enables users to perform file-transfer operations without user authentication. The use of this utility poses security issues.
<code>tftpd.pm</code>	This program module implements the <code>tftpd</code> daemon process, which services incoming Trivial File Transfer Protocol (TFTP) requests. The <code>inetd</code> daemon handles the initial client connection for the TFTP service, but then the <code>inetd</code> daemon executes the <code>tftpd.pm</code> program module and runs <code>tftpd</code> as a separate daemon, under the user <code>nobody.nobody</code> .
<code>tftpd6.pm</code>	This program module implements the <code>tftpd6</code> daemon process, which implements the server side of the Trivial File Transfer Protocol (TFTP), with support for IPv6 addresses.
<code>traceroute.pm</code>	This program module implements the <code>traceroute</code> command, which traces the route that an IP packet follows to the specified network host.
<code>xinetd.pm</code>	This program module implements the <code>xinetd</code> daemon process, a super-server that listens on many ports. It provides support for the IPv4 and IPv6 protocols.

**Table 1-5** lists the components that reside in the STCP templates subdirectory ((master\_disk)>system>stcp>templates). Use the `copy_file` command to make a copy of each template file; do **not** edit the original STCP template file. Edit the copies for your configuration and place the final versions in the ((master\_disk)>system>stcp directory so that they are available to STCP. (For a description of the `copy_file` command, see the *OpenVOS Commands Reference Manual* (R098).)

**Table 1-5. Components of the STCP Templates Subdirectory** (Page 1 of 3)

Component	Description
<code>bootptab</code>	This template file shows the format of the information required by the <code>bootpd</code> daemon process.
<code>hosts</code>	This template file shows the format of the information needed to identify hosts. In most cases, a DNS server is used. In this case, the default entries in this file should not be modified.
<code>hosts.allow</code>	This template file shows the format of information required as part of implementing TCP wrappers functionality; the <code>hosts.allow</code> file identifies clients with the service daemons that can provide access to those clients.
<code>hosts.deny</code>	This template file shows the format of information required as part of implementing TCP wrappers functionality; the <code>hosts.deny</code> lists clients with service daemons that can deny access to those clients.
<code>inetd.conf</code>	This template file shows the format of the information required by the <code>inetd</code> daemon process to handle incoming requests for services. You can configure the <code>inetd</code> daemon process to directly handle requests for services like <code>bootpd</code> and <code>tftpd</code> , or you can configure the <code>inetd</code> daemon process to invoke the <code>tcpd</code> daemon process to handle those requests.
<code>networks</code>	This template file shows the format of the information associated with network and subnetwork names and numbers.
<code>nsswitch.conf</code>	This template file shows the format for the lists of databases that are sources of information about IP addresses.
<code>omonconf</code>	This template file shows the format of the destinations file that lists the routers that you plan to monitor from this module, and that can monitor this module.
<code>ospfdconf</code>	This template file shows the format of the information needed by the <code>ospfd</code> daemon process.
<code>protocols</code>	This template file shows the format of the information needed to identify the supported protocols.

**Table 1-5. Components of the STCP Templates Subdirectory** (Page 2 of 3)

Component	Description
<code>resolv.conf</code>	This template file shows the format of the name-server information needed if you are using DNS servers for host name resolution.
<code>services</code>	This template file shows the format of the information needed to identify the services available on the module (for example, incoming services associated with <code>telnetd</code> and services such as <code>bootpd</code> and <code>tftpd</code> ).
<code>snmpconf</code>	This template file shows the format of the system-related information recognized by the <code>snmpd</code> daemon process.
<code>start_inetd.cm</code>	<p>This template command macro shows the format of the information needed to start the POSIX.1-compliant <code>inetd</code> daemon process. The user <code>root</code> must issue this command macro.</p> <p>Note that the file <code>sample_start_inetd.release_number</code> exists in the <code>(master_disk)&gt;system&gt;release_dir</code> directory. This sample file is identical to the template file.</p>
<code>start_stcp.cm</code>	<p>This template command macro shows the format of the information needed to start the various portions of STCP on the module. It enables you to bring up the STCP protocol stack, start the daemon processes (such as <code>ftpd</code>, <code>telnetd</code>, and <code>snmpd</code>), and add each logical NIC using the <code>ifconfig</code> command.</p> <p>Note that the file <code>sample_start_stcp.release_number</code> exists in the <code>(master_disk)&gt;system&gt;release_dir</code> directory. This sample file is identical to the template file.</p>
<code>start_xinetd.cm</code>	<p>This template command macro shows the format of the information needed to start the POSIX.1-compliant <code>xinetd</code> daemon process. The user <code>root</code> must issue this command macro.</p> <p>Note that the file <code>sample_start_xinetd.release_number</code> exists in the <code>(master_disk)&gt;system&gt;release_dir</code> directory. This sample file is identical to the template file.</p>
<code>stop_inetd.cm</code>	<p>This template command macro shows the format of the information needed to stop the POSIX.1-compliant <code>inetd</code> daemon process. You must be logged in as privileged in order to issue this command macro.</p> <p>Note that the file <code>sample_stop_inetd.release_number</code> exists in the <code>(master_disk)&gt;system&gt;release_dir</code> directory. This sample file is identical to the template file.</p>
<code>stop_xinetd.cm</code>	<p>This template command macro shows the format of the information needed to stop the POSIX.1-compliant <code>xinetd</code> daemon process. You must be logged in as privileged in order to issue this command macro.</p> <p>Note that the file <code>sample_stop_xinetd.release_number</code> exists in the <code>(master_disk)&gt;system&gt;release_dir</code> directory. This sample file is identical to the template file.</p>



**Table 1-5. Components of the STCP Templates Subdirectory** (Page 3 of 3)

Component	Description
<code>sync_cfgd.conf</code> <code>sync_cfgd.leases</code>	These template files show the format of the information required by the <code>sync_cfgd</code> daemon, which is a Dynamic Host Configuration Protocol (DHCP) server for network I/O enclosure devices. The <code>sync_cfgd</code> daemon automatically assigns IP addresses to these devices based on the information in the <code>sync_cfgd.conf</code> file. The <code>sync_cfgd.leases</code> file requires times in UTC (GMT), not your local timezone. For complete information, see the <i>Stratus ftServer: Network I/O Enclosure Guide</i> (R608).
<code>telnetSERVICE</code>	This template file shows the format of the information required by the TELNET server to handle incoming connections. It initially defines the default TELNET login service; you can tailor a copy of the template to define additional local (login and incoming slave) TELNET services (and define the local services in the <code>SERVICES</code> database file) before starting the <code>telnetd</code> daemon process. (Once the daemon is running, use the <code>telnet_admin</code> command to add, delete, verify, or modify the local TELNET services.)
<code>xinetd.conf</code>	This template file shows the format of the information required by the <code>xinetd</code> daemon process to handle incoming requests for services.

## Related Drivers

STCP requires the STREAMS Data-Link Multiplexer (SDLMUX) as a communications driver and a lower-level device driver, as the following sections explain:

- [“SDLMUX” on page 1-15](#)
- [“Lower-Level Device Drivers” on page 1-16](#)

## SDLMUX

The *STREAMS Data-Link Multiplexer* (SDLMUX) is an OpenVOS communications driver that provides a consistent data-link interface between STCP and the lower-level device drivers for network interface cards. *Network interface cards* (NICs) are ports on communications boards, cards, or adapters. NICs physically connect a module to a local area network (LAN).

SDLMUX manages simplex NICs and partnered NICs that are configured in logical entities called SDLMUX-group interfaces. *Partnered NICs* are two ports on adapters of the same type that work together to preserve communication after a hardware failure. A *simplex NIC* is a single port on a board, card, or adapter that functions as an independent NIC. When you configure two NICs as partners connected to the same LAN segment, SDLMUX can perform a hardware failover. A *hardware failover* is the

automatic transfer of communications input/output (I/O) from a nonoperational or faulty NIC (or a faulty connection associated with the NIC) to its operational partner.

NICs on ftServer modules are ports on Ethernet PCI adapters (see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) for detailed information about these NICs). To ensure failover if a port or adapter fails, you configure two ports on adapters of the same type as partners in an SDLMUX-group interface, where each adapter is in a different enclosure on the same module.

SDLMUX is both loadable and unloadable. You do not have to explicitly load SDLMUX since STREAMS loads it automatically when you initialize the first SDLMUX-associated NIC. SDLMUX, which is the file `sdlmux.cp.pm`, is located in the directory `(master_disk)>system>kernel_loadable_library`.

This manual provides SDLMUX-specific information in the following sections and chapters.

- “[Lower-Level Device Drivers](#)” on page 1-16 describes the lower-level device drivers that SDLMUX requires
- “[Configuring SDLMUX Devices](#)” on page 3-12 describes how to configure SDLMUX.
- The `dlnux_admin` command description explains how to administer SDLMUX using the `dlnux_admin` command.

## Lower-Level Device Drivers

SDLMUX requires a STREAMS lower-level device driver and its associated Ethernet PCI adapters. Like SDLMUX, each device driver is loaded automatically by STREAMS; you do not need to explicitly load them. Also like SDLMUX, these device drivers are located in the `(master_disk)>system>kernel_loadable_library` directory.

The lower-level device drivers are as follows:

- IGB (`igb.cp.pm`)
- IXGBE (`ixgbe.cp.pm`)

For information on the lower-level device driver that is required for a specific Ethernet PCI adapter, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679), which also provides information about the PCI adapters, including the ftServer systems that support them.

## Hardware Requirements

The hardware requirement of STCP is a physical network interface to a LAN. On ftServer modules, ports on Ethernet PCI adapters (with their cables and connectors) are the NICs for STCP. To configure an Ethernet PCI adapter, you must create an entry for each port in the `devices.tin` file. In order for STCP to recognize and manage these ports, you must also configure the ports (simplex and duplexed) in an SDLMUX-group interface.

For detailed information about the Ethernet PCI adapters, their ports, and how to configure them, see the following documentation:

- The *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) describes the Ethernet PCI adapters that ftServer modules support.
- [“Planning the SDLMUX Configuration” on page 3-3](#) describes how to plan the physical and logical configuration of the ports on the adapters.
- [“Configuring SDLMUX Devices” on page 3-12](#) describes how to create `devices.tin` file entries for these ports and how to configure them in an SDLMUX-group interface.

## Software Requirements and Considerations

STCP is bundled with OpenVOS on the OpenVOS release media; it is not a separate product. STCP requires the installed OpenVOS release running on a supported hardware platform. In addition, the version of STCP that this manual describes requires the following software.

- The OpenVOS device configuration-table file (`devices.table`) is installed on your system in the `(master_disk)>system` directory; its `devices.tin` file exists in the `(master_disk)>system>configuration` directory. These files should have entries for each NIC and SDLMUX-group interface as well as entries for the STCP devices. If you need to change your configuration, or if the `devices.tin` file does not contain the necessary entries, you need to create them. (See [Chapter 3](#) and [Chapter 4](#).)
- To ensure that OpenVOS loads the related communications and device drivers, you must check that the `start_stcp.cm` file has the appropriate `dlnux_admin device_name init_sdlnux` command lines activated. (See [Chapter 6](#).)
- To enable the module to run STCP, you must explicitly load several STCP components from the `module_start_up.cm` file. (See [Chapter 6](#) for more information.)
- To use incoming STCP TELNET, you **must** run the STCP TELNET server ([telnetd](#)). Because STCP TELNET relies on the window terminal software, the TELNET login and slave devices are classified and listed as window terminal devices.

- STCP provides a C-language application programming interface (API). If you want to use a different programming language, you must write your own wrapper file.

Before configuring the STCP software, you should consider the following administrative requirements.

- By default for IPv4, STCP enables the module to act as a router for IPv4 and to forward packets. This feature is called *IP forwarding*. It is relevant only if you have multiple SDLMUX-group interfaces running STCP; if you have only a single SDLMUX-group interface running STCP, IP forwarding has no meaning and is effectively disabled. If you do **not** want the module to act as a router when you have multiple NICs connected to different networks/subnets, you must explicitly disable IP forwarding. (The `IP_forwarding` command displays or changes the setting for IP forwarding.) Note that you **cannot** selectively disable/enable IP router/gateway functions on individual NICs; the setting affects all interfaces configured for STCP on the module. Stratus recommends that you disable IP forwarding.
- The SNMP variables that are maintained by the STCP SNMP daemon (the `snmpd` daemon process) can be viewed (or potentially altered) by a remote SNMP manager (an SNMP client). For example, a remote SNMP manager whose community is set to private can control the IP forwarding feature by manipulating the `ipForwarding` variable. (See the description of `snmpd` for a description of SNMP variables that can be set.)

---

## Chapter 2

# Configuring STCP, OSPFD, and STCP TELNET

This chapter summarizes the procedures required to initially configure STCP on an OpenVOS module. This chapter also summarizes the procedures required to configure the STCP implementation of the TELNET server and OSPFD. This chapter contains the following sections.

- [“Checklist for Configuring STCP” on page 2-1](#)
- [“Checklist for Configuring the TELNET Daemon” on page 2-5](#)
- [“Checklist for Configuring the OSPFD Daemon Process” on page 2-7](#)
- [“Checklist for TCP Wrappers Functionality” on page 2-7](#)
- [“Checklist for Enabling IPv6 Support” on page 2-9](#)

This chapter is intended to serve as a configuration checklist for those generally familiar with OpenVOS configuration procedures. If you are unfamiliar with OpenVOS configuration procedures, see the OpenVOS System Administration manuals listed in the Preface.

### NOTE

To perform the procedures mentioned in this chapter, you must be a privileged user and have modify access to the (master\_disk)>system>configuration and (master\_disk)>system>stcp directories. (The manual *OpenVOS System Administration: Registration and Security* (R283) provides information about establishing user access rights.)

## Checklist for Configuring STCP

STCP software should already be installed on your module. Required Ethernet PCI adapters should already be installed and the ports configured. OpenSSL and OpenSSH are recommended for your module. If this work is already complete, proceed with step 6; otherwise, begin with step 1.

To configure STCP on your module, perform the following steps.

1. Plan the physical configuration of Ethernet PCI adapters and the logical configuration of SDLMUX-group interfaces. Then, install the required hardware on the module and establish the network connections. See the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) and [Chapter 3](#).

Step complete: ☐

2. Install OpenSSL and configure OpenSSH (recommended). See *Software Release Bulletin: Internet Security Pack for OpenVOS, Release 4.0.2* (R660).

Step complete: ☐

3. Create `devices.tin` entries for the ports on the Ethernet PCI adapters and the SDLMUX-group interfaces. For information about creating these entries, see [Chapter 3](#).

Step complete: ☐

4. Create all of the appropriate `devices.tin` entries for the STCP protocol drivers, TELNET devices, and related components (for example, device entries for the network interfaces). For information about how to create the STCP device entries, see [Chapter 4](#). (The next section summarizes TELNET configuration procedures.)

Step complete: ☐

5. Re-create the `devices.table` file in the `(master_disk)>system>configuration` directory, and then copy the file to the `(master_disk)>system` directory on all modules in the system, including the current module. (For more information about handling configuration-table files, see the manual *OpenVOS System Administration: Configuring a System* (R287).)

Step complete: ☐

6. Create your own database files and start STCP command macro using the template files (the STCP database files are described in [Chapter 5](#) and the command macros `start_stcp.cm`, `start_inetd.cm`, `start_xinetd`, `stop_xinetd`, and `stop_inetd.cm` are described in [Chapter 6](#)).

- a. Add information to the database files that is accurate for your configuration. For example, the `telnetSERVICE` database file must define the TELNET information for permanent local services. (The next section summarizes TELNET configuration procedures.)

Place your own database files in the `(master_disk)>system>stcp` directory; leave the original template files unmodified in the `(master_disk)>system>stcp>templates` directory.

- b. Add information to the start STCP command macro that is accurate for your configuration. For example, the `start_stcp.cm` command macro must contain an `ifconfig` command to define each logical network interface to the stack. It must also contain the appropriate `dlnux_admin device_name`

`init_sdlmux` command lines. In the `start_stcp.cm` command macro, you should also uncomment and, if necessary, modify the command lines to start the daemon processes that your modules require. (The `start_inetd.cm`, `stop_inetd.cm`, `start_xinetd`, and `stop_xinetd` command macros typically do not require modification.)

Place your final versions of the `start_stcp.cm`, `start_inetd.cm`, and `stop_inetd.cm` command macros in the `(master_disk)>system>stcp` directory.

Step complete: ☐

7. Make sure that the `module_start_up.cm` file activates the following:
  - a. the STCP library paths—For information about establishing the STCP command library, see the section “[Establishing the STCP Command Library Paths](#)” on page 6-3.
  - b. the module’s host name—For more information about establishing the module’s STCP host name, see the section “[Establishing the Module’s STCP Host Name](#)” on page 6-4.
  - c. the `start_stcp.cm` command macro—For information about modifying the `module_start_up.cm` file and the contents of the `start_stcp.cm` command macro, see the section “[Modifying the module\\_start\\_up.cm and start\\_stcp.cm Files](#)” on page 6-3.

#### NOTE

The `start_stcp.cm` command macro enables you to specify `route` commands to define host, network, or default routes. The template macro shows a sample `route` command that is commented out. Make sure that you specify your own `route` commands. Your `route` commands must appear **after** the appropriate `ifconfig` commands in the command macro.

Step complete: ☐

8. Unless you need to reboot the module, perform steps 9 through 13. (If you need to reboot, see the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).)  
Step complete: ☐
9. Issue the `hostname` command with the host name of the module.  
Step complete: ☐
10. Issue the `configure_devices` command to activate the changes you made to the `devices.table` file earlier in this procedure. See the manual *OpenVOS System Administration: Configuring a System* (R287) for more information about `configure_devices`.  
Step complete: ☐
11. Use STCP security-related parameters. (See [Chapter 7](#).)  
Step complete: ☐
12. Start STCP by invoking the `start_stcp.cm` command macro. This command macro loads most of the STCP components and should contain your own `ifconfig` commands to add the logical network interfaces and your own `dldmux_admin device_name init_sdlmux` command lines to add the physical network interfaces. (See [Chapter 9](#).) If you decided not to place `route` commands in the command macro but still need to define routes, issue the appropriate `route` commands.  
Step complete: ☐
13. If the `module_start_up.cm` command macro invokes the `start_inetd.cm` command macro, start the `inetd` daemon process by invoking the `start_inetd.cm` command macro. If the `start_stcp.cm` command macro invokes the `start_inetd.cm` command macro, do not perform this step.  
Step complete: ☐
14. Verify that the STCP components have been initialized correctly, using commands such as `list_library_paths`, `list_devices`, `netstat` (with the `-protocol` argument), and `ifconfig` (with the `-all` argument). (For more information about verifying aspects of STCP, see [Chapter 6](#).)  
Step complete: ☐



## Checklist for Configuring the TELNET Daemon

To initially configure the `telnetd` daemon process on your module, perform the following steps.

### NOTE

The standard STCP TELNET server (started by the `telnetd` command) does not support the IPv6 protocol; instead, it supports only IPv4. Moreover, the TELNET protocol is non-secure. To increase security and/or to accommodate incoming login requests and incoming slave requests with IPv6 protocol support, use the secure shell (SSH) utility. For information on SSH, see the *Software Release Bulletin: Internet Security Pack for OpenVOS, Release 4.0.2 (R660)*.

1. Create the appropriate TELNET device entries in the `devices.tin` file. You must create device entries for the following:
  - the module's `tli_al` access layer—For more information, see the section [“Defining the TLI Access-Layer Device Driver” on page 4-9](#).
  - the two TELNET pipe devices—For more information, see the section [“Defining STCP TELNET Pipe Entries” on page 4-10](#).
  - any login, incoming slave, and outgoing slave devices—For more information, see the section [“Defining STCP TELNET Devices” on page 4-8](#).

Step complete: ☐

2. Re-create the `devices.table` file in the `(master_disk)>system>configuration` directory, and then copy the file to the `(master_disk)>system` directory on all modules in the system, including the current module. (For more information about handling configuration-table files, see the manual *OpenVOS System Administration: Configuring a System (R287)*.)

Step complete: ☐

3. Issue the `configure_devices` command to activate the changes you made to the `devices.table` file earlier in this procedure. See the manual *OpenVOS System Administration: Configuring a System (R287)* for more information about `configure_devices`.

Step complete: ☐

4. STCP's default TELNET login service is defined in the `telnet-service` database file. Edit your own `telnet-service` file to add login services, change the existing service, or add any incoming slave services. (Outgoing slave services are not defined in the local `telnet-service` file.) For more information, see the section "[The telnet-service File and TELNET Information](#)" on page 5-53.

Step complete: ☐

5. Edit your own `services` database file if you have added additional TELNET services to the `telnet-service` file. (For more information, see the section "[The services File and Service Information](#)" on page 5-46.)

Step complete: ☐

6. Place your versions of the `telnet-service` and `services` database files in the `(master_disk)>system>stcp` directory.

Step complete: ☐

7. Once you have configured the rest of STCP (as described in the previous section and in [Chapter 6](#)), execute the `start_stcp.cm` command macro, which starts `telnetd` as a privileged process.

Step complete: ☐

While the `telnetd` daemon process is running, you can use the `telnet_admin` command to add, delete, modify, or verify local TELNET services. [Chapter 9](#) describes the `telnet_admin` command.

#### NOTE \_\_\_\_\_

STCP TELNET uses several program modules (`telnet.pm`, `telnetd.pm`, and `telnet_admin.pm`, which reside in the `(master_disk)>system>stcp>command_library` directory) and STREAMS drivers (`tpipe.cp.pm` and `tnmod.cp.pm` located in the directory `(master_disk)>system>kernel_loadable_library`). If you have problems making TELNET operational, make sure that these software components have not been deleted or moved from their appropriate directories.

## Checklist for Configuring the OSPFD Daemon Process

To configure and start the `ospfd` daemon process on your module, perform the following steps.

1. Use the template file to create an `ospfdconf` database file. The template `ospfdconf` file resides in the `(master_disk)>system>stcp>templates` directory. In a copy of this template file, enter information according to the template format. When you have finished creating your own version of the `ospfdconf` file, place the version in the `(master_disk)>system>stcp` directory. STCP must be able to find the `ospfdconf` file in this directory.

For more information about this file, see [“The `ospfdconf` File and Routing Information” on page 5-31](#).

Step complete: ☐

2. To automatically start the `ospfd` daemon whenever the system restarts, remove the ampersand (&) and space from the OSPF entries in the `start_stcp.cm` file.

For more information about OSPF entries, see the [`ospfd` command description in Chapter 8](#).

Step complete: ☐

3. Issue the `ospfd` command. To run the `ospfd` daemon as a background process, use the display form of the command.

For information about the `ospfd` command, see the [`ospfd` command description in Chapter 8](#).

Step complete: ☐

While the [`ospfd`](#) daemon process is running, you can use the `omon` command to monitor other OSPF routers. [Chapter 9](#) describes the `omon` command.

## Checklist for TCP Wrappers Functionality

To configure TCP wrappers functionality, perform the following steps.

1. When binding a TCP wrappers application, include the correct version of the object library. STCP includes two versions of an API for TCP wrappers functionality:
  - A legacy version of the TCP wrappers library—When binding a legacy TCP wrappers application, include the following directory among the object library directories searched by the binder:  
`(master_disk)>system>stcp>object_library>tcpwrappers`
  - An IPv6-enabled version—When binding an application for IPv6 protocol support, include the following directory, as with any POSIX application, among

the object library directories searched by the binder:

```
(master_disk)>system>posix_object_library
```

You can bind a TCP wrappers application with only one version of the object library. The correct version is bound in automatically.

Step complete: ☐

2. When you start the `ftpd` and the `telnetd` daemon processes, you must specify the value `yes` for the `-tcpwrapper_check` arguments. You can also specify values for the `-numeric` arguments when you specify values for the `-tcpwrapper_check` arguments.

For more information about these arguments, see the descriptions of the `ftpd` and `telnetd` commands in [Chapter 8](#).

Step complete: ☐

3. To enable TCP wrappers functionality in a service that the `inetd` daemon starts, the `inetd` daemon must first invoke the `tcpd` daemon to start the service rather than the `inetd` daemon invoking the service directly. To enable the `inetd` daemon to first invoke the `tcpd` daemon to start the service, you must modify the service's entry in the `inetd.conf` file. (The `xinetd` daemon does not use TCP wrappers functionality.)

For more information about the `inetd.conf` file, see [“The `inetd.conf` File and Information for `inetd`” on page 5-21](#).

Step complete: ☐

4. In the `hosts.allow` file, you must list the path names of the daemons for services to which you want to provide client access; you must also list the names of the clients to which you want to give access. In the `hosts.deny` file, you must list the names of clients and the path names of the daemons for services to which you want to deny client access.

For more information about the `hosts.allow` and `hosts.deny` files, see [“The `hosts.allow` and `hosts.deny` Files and TCP Wrappers” on page 5-10](#).

Step complete: ☐

## Checklist for Enabling IPv6 Support

To enable STCP to support IPv6 on your module, perform the following steps.

1. When you configure interfaces using the `ifconfig` command and static routes using the `route` command, you simply specify an IPv6 address.

For IPv6 support, you do not configure a default IPv6 route because the protocol determines it. If the module is connected to more than one router, you can use the `ifconfig` command and/or router configuration to modify the way the IPv6 protocol determines the default route.

For information about the `ifconfig` and `route` commands, see [Chapter 9](#).

Step complete: ☐

2. Check that the following command line in the `start_stcp.cm` file is uncommented:

```
(master_disk)>system>command_library>sleep -seconds
+(calc 1 + (system_info 1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.
+1.4.109))
```

This command line ensures that duplicate address detection (DAD) finishes after adding all of the interfaces. To uncomment a command line, delete the ampersand and space (&) at the beginning of the line.

For information about the `start_stcp.cm` file, see “[Modifying and Executing the start\\_stcp.cm Command Macro](#)” on page 6-4.

Step complete: ☐

3. Check that the `start_stcp.cm` file sets the `ipv6_devs` variable to contain each interface configured for IPv6 support.

The `ipv6_devs` variable, which is in the `set_string ipv6_devs` command line, defines the list of interfaces that the `rtsold` daemon process operates on. Therefore, the command line should contain all of the interfaces that are enabled for IPv6 by the preceding `ifconfig` command lines (that is, the `ifconfig` command lines preceding the `set_string ipv6_devs` command line in the `start_stcp.cm` file) *except* for interfaces that will be configured for some other IPv6 router.

Modify the following command lines, as necessary:

```
& *****RTSOLD*****
& rtsold should be started on all interfaces which are configured for
IPv6
!display_line Starting rtsold .....
```

*(Continued on next page)*

```
&
&set_string rtsold_cmd (master_disk)>system>stcp>command_library>rtsold
&set_string ipv6_devs #sdlmux.10.7.0 #sdlmux.10.7.3 #sdlmux.10.8.0 &+
                        #sdlmux.10.8.1 #sdlmux.10.8.2
&set_string rtsold_args -foreground
start_process -output_path (master_disk)>system>stcp>logs>rtsold.out &+
    (string &rtsold_cmd& &ipv6_devs& &rtsold_args&) &+
    -privileged -process_name rtsold
```

For information about the `start_stcp.cm` file, see [“Modifying and Executing the start\\_stcp.cm Command Macro” on page 6-4](#).

Step complete: ☐

---

## Chapter 3

# Configuring SDLMUX

This chapter, which contains the following sections, describes how to configure SDLMUX.

- “[Overview of Device Configuration](#)” on page 3-1
- “[Planning the SDLMUX Configuration](#)” on page 3-3
- “[Sample SDLMUX Configuration](#)” on page 3-7
- “[Software Configuration Procedures for SDLMUX](#)” on page 3-10
- “[Configuring SDLMUX Devices](#)” on page 3-12
- “[Initializing an SDLMUX-Group Interface](#)” on page 3-21
- “[Loading and Unloading SDLMUX](#)” on page 3-23
- “[Defining the SDLMUX-Group Interfaces](#)” on page 3-25
- “[Changing Existing Configurations](#)” on page 3-26
- “[SDLMUX Requests of the `analyze\_system` Subsystem](#)” on page 3-31

### Overview of Device Configuration

To configure SDLMUX, you configure ports on Ethernet PCI adapters as well as their SDLMUX-group interfaces. This chapter explains how to plan the SDLMUX configuration as well as how to configure the required devices. [Chapter 4](#) explains how to configure additional required devices for STCP. For information on the Ethernet PCI adapters, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

As part of any OpenVOS software configuration process, you must configure devices in the `devices.tin` file and then from it, create the device configuration-table file (`devices.table`). The `devices.table` file enables OpenVOS to recognize all of the devices within a system of Stratus modules.

The `devices.table` file defines a device as either a hardware or software entity that the system and its users can reference and access. Communications products typically introduce unique sets of protocol-specific devices in the `devices.tin` and `devices.table` files.

Like all other configuration-table files, OpenVOS compiles the `devices.table` file from a pair of files: a data description file (`devices.dd`) and its corresponding table input file (`devices.tin`).

- The `devices.dd` file is a format file that declares the template of records in the `devices.table` file. You never modify this file; you only display it to see which fields you must use in a record (entry) for an STCP device.
- The `devices.tin` file is a text file in which you create an entry for each device. This file contains additional fields that apply to other types of devices; for a complete list of these fields, see the manual *OpenVOS System Administration: Configuring a System* (R287).

The `devices.table`, `devices.dd`, and `devices.tin` files reside in the `(master_disk)>system>configuration` directory of each module. You typically edit the `devices.tin` file that is located in the directory `(master_disk)>system>configuration` of the master module.

After you define the required devices in the `devices.tin` file, you must convert the configuration information to a machine-executable file called `devices.table`. Issue the `create_table` command from the directory `(master_disk)>system>configuration` to generate a new `devices.table` file, as follows:

```
create_table devices
```

After you have re-created the `devices.table` file, you must install it in the `(master_disk)>system` directory on all modules in the system (including the current module). To do so, issue the following `broadcast_file` command.

```
broadcast_file devices.table (master_disk)>system
```

The newly re-created `devices.table` file takes effect at the next bootload, when the `module_start_up.cm` file executes the `configure_devices` command.

For descriptions of the `create_table`, `broadcast_file`, and `configure_devices` commands, see the manual *OpenVOS System Administration: Configuring a System* (R287).



## Planning the SDLMUX Configuration

You should first plan the SDLMUX configuration before you begin modifying files and/or issuing commands to create the configuration. You must first plan the network topology, the physical configuration, and then the logical configuration in addition to considering software requirements, as the following sections describe.

- [“Planning the Physical Configuration for SDLMUX” on page 3-3](#)
- [“Planning the Logical Configuration of SDLMUX-Group Interfaces” on page 3-6](#)

## Planning the Physical Configuration for SDLMUX

Planning the physical configuration includes planning the location of the Ethernet PCI adapters in your module. (For information about PCI adapters as well as their slots and enclosures, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).)

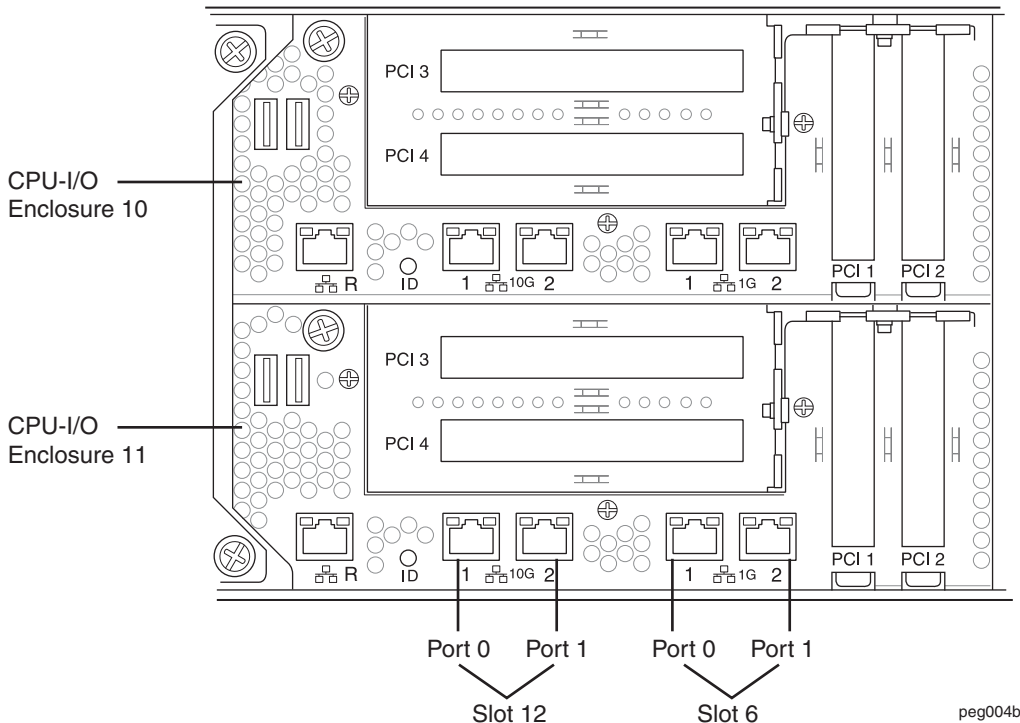
Each enclosure with PCI-adapter slots also contains one or two (depending on the system) embedded 10/100/1000-Mbps Ethernet PCI adapters and three slots for PCI adapters that you can insert. The enclosures are numbered 10 and 11. The following sections describe the PCI-adapter slot numbers of ftServer systems:

- [“PCI-Adapter Slot Numbers of ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 Systems” on page 3-3](#)
- [“PCI-Adapter Slot Numbers of ftServer V 2404, V 4408, V 6408, and V 6512 Systems” on page 3-5](#)

### PCI-Adapter Slot Numbers of ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 Systems

For PCI-adapter slots on ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 systems, OpenVOS requires the PCI-slot numbers, which are the values 1, 2, 3, and 4. The OpenVOS slot number for the embedded 1Gb Ethernet PCI adapters is 6. The OpenVOS slot number for the embedded 10Gb Ethernet PCI adapters is 12. For each pair of embedded Ethernet PCI adapters in an enclosure, looking at the rear of the enclosure, the left connector is port 0; the right connector is port 1.

[Figure 3-1](#) illustrates the numbering of PCI-adapter slots on ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 systems (the locations or existence of other features such as USB ports may vary on different V Series models).



**Figure 3-1. PCI-Adapter Slots on ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, V 6832 Systems**

As you plan the physical configuration of Ethernet PCI adapters on these systems, you must consider the following:

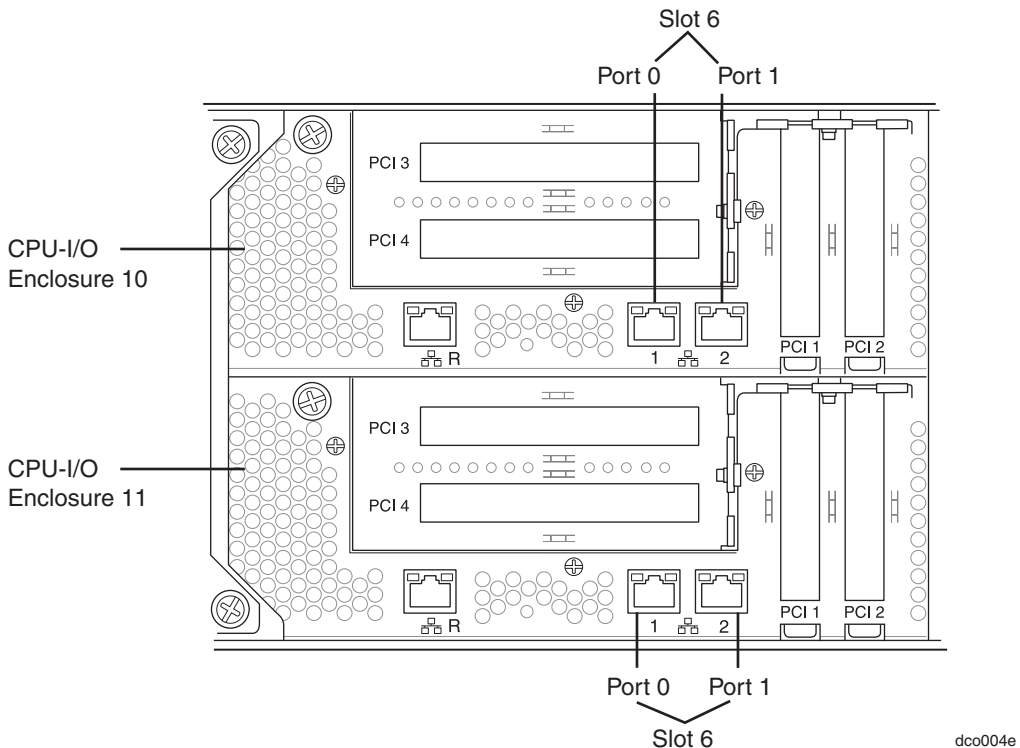
- In each CPU-I/O enclosure, slot number 1 is reserved for the PCI adapter that connects to a storage enclosure.
- Embedded Ethernet PCI adapters 10/6/0 and 11/6/0 are typically reserved for the Ethernet switch.

You need to determine which ports will be partnered in one SDLMUX-group interface and which (if any) will be simplexed. In addition, you must observe the configuration requirements of SDLMUX-group interfaces and other software configuration requirements of Ethernet PCI adapters, as described later in this chapter.

### PCI-Adapter Slot Numbers of ftServer V 2404, V 4408, V 6408, and V 6512 Systems

For PCI-adapter slots on ftServer V 2404, V 4408, V 6408, and V 6512 systems, OpenVOS requires the PCI-slot numbers, which are the values 1, 2, 3, and 4. The OpenVOS slot number for the embedded Ethernet PCI adapters is 6. In each enclosure, the left embedded Ethernet PCI adapter is port 0; the right embedded Ethernet PCI adapter is port 1.

Figure 3-2 illustrates the numbering of PCI-adapter slots on ftServer V 2404, V 4408, V 6408, and V 6512 systems.



**Figure 3-2. PCI-Adapter Slots on ftServer V 2404, V 4408, V 6408, and V 6512 Systems**

As you plan the physical configuration of Ethernet PCI adapters on these systems, you must consider the following.

- In each CPU-I/O enclosure, slot number 1 is reserved for the PCI adapter that connects to a storage enclosure.
- Embedded Ethernet PCI adapters are typically reserved for the network I/O enclosure and the Ethernet switch.

You need to determine which ports will be partnered in one SDLMUX-group interface and which (if any) will be simplexed. In addition, you must observe the configuration requirements of SDLMUX-group interfaces and other software configuration requirements of Ethernet PCI adapters, as described in the remainder of this chapter.

## Planning the Logical Configuration of SDLMUX-Group Interfaces

You create a logical configuration by configuring SDLMUX-group interfaces. An SDLMUX-group interface typically consists of two ports on two different Ethernet PCI adapters. The SDLMUX-group interface partners the adapters to create fault-tolerant network interfaces (that is, duplexed or grouped in a pair). When you plan the configuration of SDLMUX-group interfaces, you must ensure the following requirements.

- Two ports must be configured in the same logical SDLMUX-group interface.
- Two ports must have the same MTU.
- Two ports must be connected to the same logical TCP/IP subnetwork and be able to exchange SDLMUX test frames containing a nonstandard upper-layer protocol, in order to support seamless failover and continued communication in the event of a hardware or network failure. (Note that for configuration purposes, network interfaces that are partnered in one SDLMUX-group interface share one IP address, while nonpartnered network interfaces have unique IP addresses.)

### NOTE

A simplexed SDLMUX-group interface contains only one port. This type of SDLMUX-group interface is not fault-tolerant. You must configure a simplexed port in an SDLMUX-group interface in order for the port to handle IP multicasting.

In addition to the requirements above for SDLMUX-group interfaces, you must adhere to the following requirements for adapters:

- The two ports in the SDLMUX-group interface must be on adapters that are of the same hardware type (for example, two embedded Ethernet PCI adapters).
- The two adapters in the SDLMUX-group interface must be located in partner enclosures on the same module and in the same numbered slot. You cannot pair two adapters that are in the same enclosure.

Additionally, you should adhere to the recommendation that the two ports in the SDLMUX-group interface should be the same port number or letter (for example, the two ports should be port A, which is port 0 in the software).

The requirements and recommendation above exist for simplicity and because the performance of PCI slots differs. If you do not follow the requirements and recommendation above, the operating system may reject the configuration or the performance of the network connection may vary, depending on which adapter is active, and a failover may change system performance in an undesirable way.

See “[Sample SDLMUX Configuration](#)” on page 3-7 for an example of an SDLMUX configuration with four SDLMUX-group interfaces.

## Sample SDLMUX Configuration

This sample SDLMUX configuration is based on [Figure 3-3](#), which shows an ftServer V 2404, V 4408, V 6408, or V 6512 system. For information on the PCI-adapter slot numbering of these systems, see “[PCI-Adapter Slot Numbers of ftServer V 2404, V 4408, V 6408, and V 6512 Systems](#)” on page 3-5.

In the sample configuration, the names of the Ethernet PCI adapters begin with `enetLAN`, where `LAN` represents the Ethernet LAN to which the PCI adapters are attached. The remainder of the name has the format `module_n.enclosure_n-slot_n.port_n`, which provides the following information:

- `module_n` is the module number, as configured by the system administrator.
- `enclosure_n` is the enclosure number. Values are 10 and 11.
- `slot_n` is the slot number. Values are 1, 2, 3, 4, and 6.
- `port_n` is the port number. For embedded Ethernet PCI adapters, values are 0 and 1. For Ethernet PCI adapters that you can insert, values are 0, 1, 2, and 3.

The names of the SDLMUX-group interfaces begin with `sdlmuxLAN`, where `LAN` represents the Ethernet LAN to which the interface is attached. The remainder of the name has the following format:

`module_n.enclosure_n-slot_n.port_n.enclosure_n-slot_n.port_n`

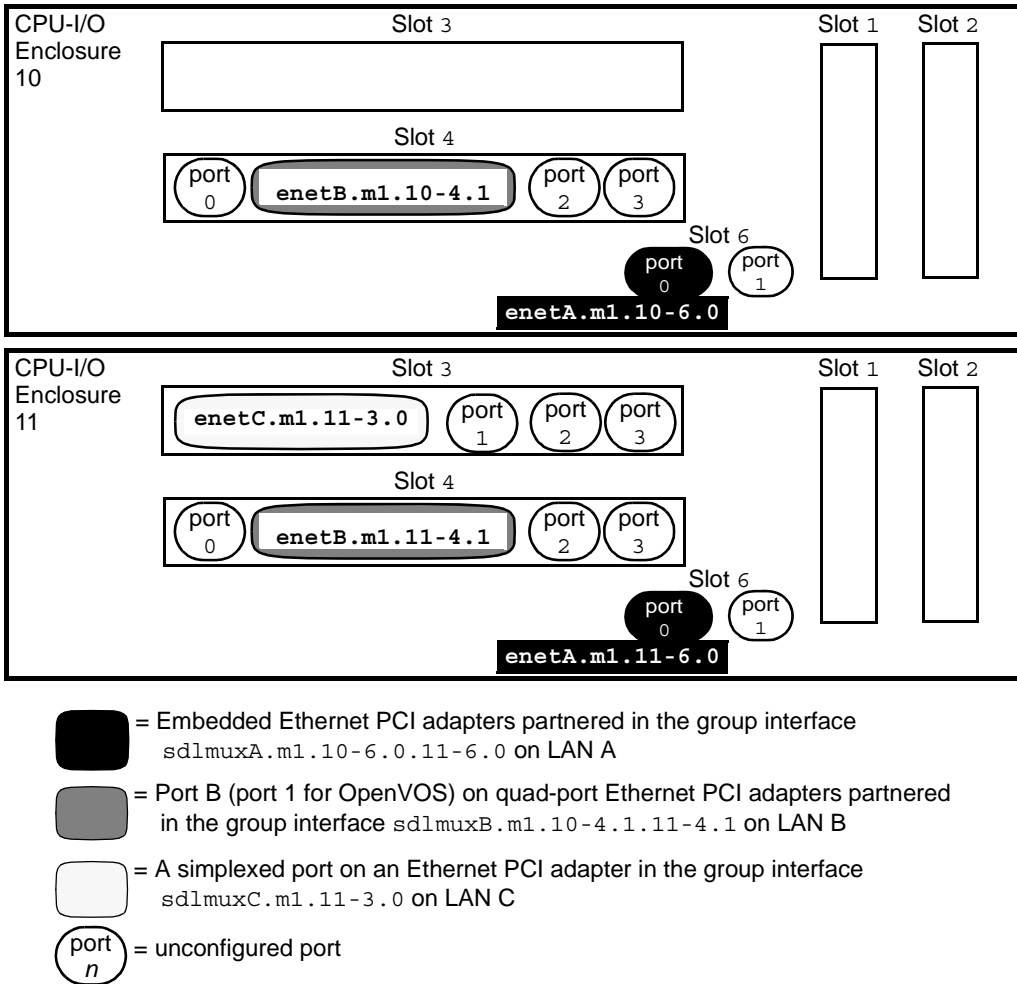
The format provides the following information:

- `module_n` is the module number (as in the name of the Ethernet PCI adapters in the interface).
- `enclosure_n-slot_n.port_n` is the enclosure number, slot number, and port number of each PCI adapter in the interface (as in the name of each Ethernet PCI adapter in the SDLMUX-group interface).

Figure 3-3 consists of the following partnered and simplex ports on Ethernet PCI adapters.

- Two partnered ports on embedded Ethernet PCI adapters (slot 5, port 0) in the two CPU-I/O enclosures (CPU-I/O enclosure 10 and 11) on m1, which are connected to the same physical Ethernet network, LAN A. These ports are named `enetA.m1.10-6.0` and `enetA.m1.11-6.0` and are configured in SDLMUX-group interface `sdlmuxA.m1.10-6.0.11-6.0`.
- Two partnered ports (port 1) on Ethernet PCI adapters in slot 10 of each CPU-I/O enclosure (CPU-I/O enclosure 10 and 11) on m1, which are connected to LAN B. These ports are named `enetB.m1.10-4.1` and `enetB.m1.11-4.1` and are configured in SDLMUX-group interface `sdlmuxB.m1.10-4.1.11-4.1`.
- One simplex port (port 0) on an Ethernet PCI adapter in slot 11 of CPU-I/O enclosure 11, which is connected to LAN C. This port is named `enetC.m1.11-3.0` and is configured in SDLMUX-group interface `sdlmuxC.m1.11-3.0`.

Figure 3-3 illustrates a sample SDLMUX configuration in an ftServer V 2404, V 4408, V 6408, or V 6512 system.



**Figure 3-3. Sample SDLMUX Configuration of Ethernet PCI Adapters in an ftServer V 2404, V 4408, V 6408, or V 6512 System**

## Software Configuration Procedures for SDLMUX

After you plan the configuration, configure the software by performing the following steps:

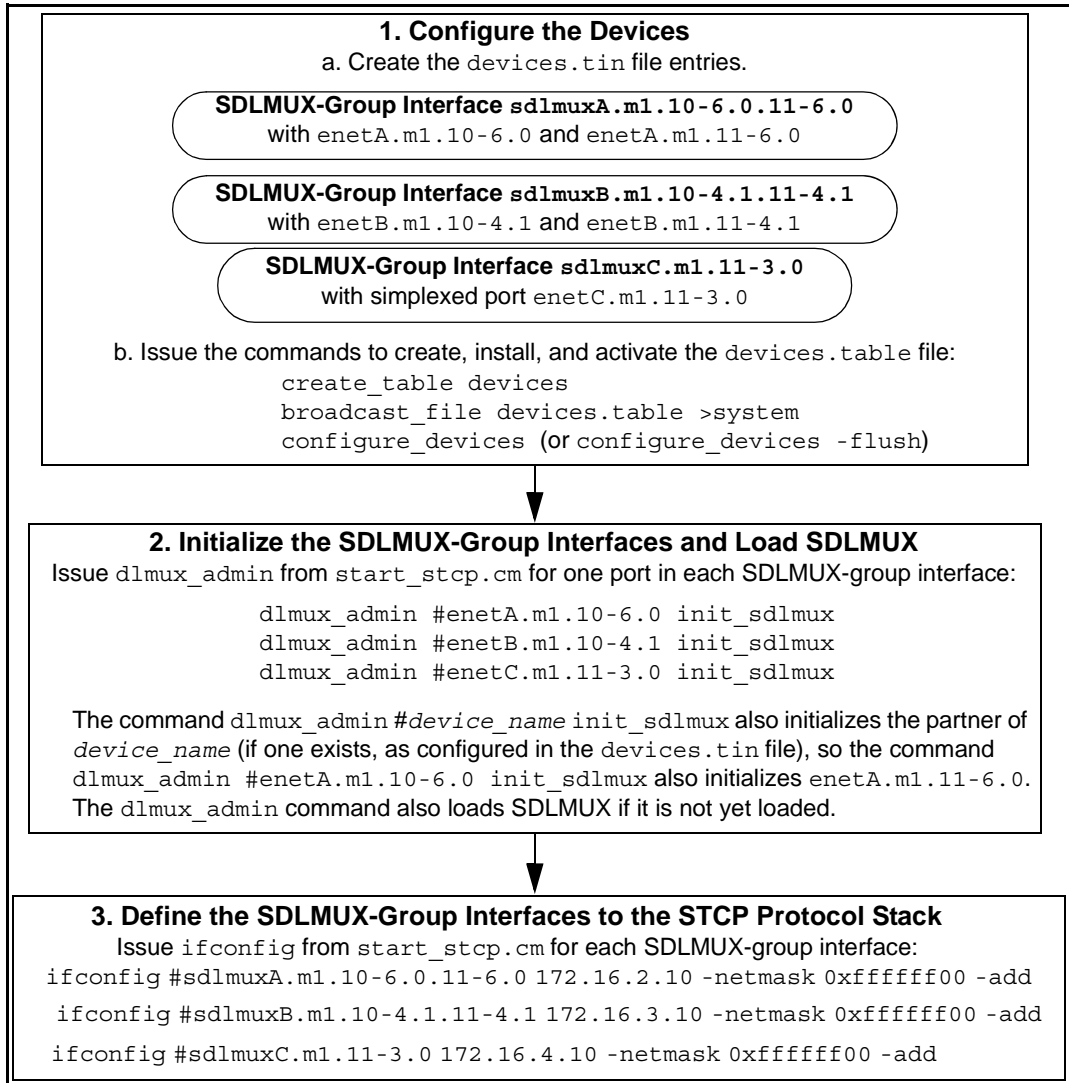
1. Configure the devices.
  - a. Create the following entries in the `devices.tin` file for the SDLMUX-group interfaces and the ports ([“Configuring SDLMUX Devices” on page 3-12](#) describes how to make these entries).
    - one entry for each SDLMUX-group interface
    - one entry for each port in the SDLMUX-group interface—This entry must include the `-sdlmux` argument of the `parameters` field in order to identify the name of the associated SDLMUX-group interface. If the port has a partner, the entry must also include the `-partner` argument in order to identify the associated partner.
  - b. After you create the device entries, you need to issue the commands to create, install, and activate the `devices.table` file. [“Overview of Device Configuration” on page 3-1](#) describes these commands.
2. Initialize each SDLMUX-group interface using the command `dlnux_admin #device_name init_sdlmux`. This command initializes the port specified in the `device_name` argument and its partner (if one exists as configured in `devices.tin` file entries) as the logical network interface for the physical network interface (that is, for the ports). This command also enables STREAMS to load SDLMUX (`sdlmux.cp.pm`) if it is not already loaded. Issue this command for only **one** port in **each** SDLMUX-group interface. You **must** issue this command **before** you issue the `ifconfig` commands described in step 3. [“Initializing an SDLMUX-Group Interface” on page 3-21](#) describes these tasks.

You can initialize an SDLMUX-group interface even if one of the adapters in the pair is broken or not present (the port must have a `devices.tin` file entry). In such cases, the `dlnux_admin` command issues a warning.
3. Define the SDLMUX-group interfaces to the STCP protocol stack using the STCP `ifconfig` command. [“Defining the SDLMUX-Group Interfaces” on page 3-25](#) describes this task.

After you configure two physical network interfaces as partners in one SDLMUX-group interface, you administer their partnering status with the `dlnux_admin` command. Remember that when you use the STCP `ifconfig` command to change the operational state of a partnered network interface, the change affects the SDLMUX-group interfaces and thus both adapters in the group.



Figure 3-4 illustrates these procedures with the commands for the configuration in “Sample SDLMUX Configuration” on page 3-7.



**Figure 3-4. SDLMUX Software Configuration Procedures**

## Configuring SDLMUX Devices

You configure devices for SDLMUX by creating a `devices.tin` file entry for each SDLMUX-group interface. You must also create a `devices.tin` file entry for each port on an Ethernet PCI adapter in order to define it as a network interface and to associate it with an SDLMUX-group interface. After you create the device entries, you issue commands to create, install, and activate the `devices.table` file. (You can find sample entries in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.)

The following sections describe how to configure SDLMUX devices.

- [“Creating Device Entries for SDLMUX-Group Interfaces” on page 3-12](#)
- [“Creating Device Entries for Ports on Ethernet PCI Adapters” on page 3-14](#)
- [“Creating, Installing, and Activating the `devices.table` File” on page 3-20](#)
- [“Verifying Devices” on page 3-20](#)

### Creating Device Entries for SDLMUX-Group Interfaces

You must create `devices.tin` file entries for each SDLMUX-group interface. The following sections describe these entries.

- [“Field Descriptions for SDLMUX-Group Interfaces” on page 3-12](#)
- [“Entries for SDLMUX-Group Interfaces” on page 3-13](#)

#### Field Descriptions for SDLMUX-Group Interfaces

The following `devices.tin` file fields are required to define SDLMUX-group interfaces.

- |   |                 |
|---|-----------------|
| ▶ <b>name</b>   | <b>Required</b> |
| <p>Specify the name of the SDLMUX-group interface. Each name must be unique, and all names are arbitrary. The name must not exceed 30 characters. By convention, the name is <code>sdlmux.device_names_suffix</code>, where <code>device_names_suffix</code> is the suffix (indicating the location of the ports in the interface) of the device names of the ports in the SDLMUX-group interface. For example, if the SDLMUX-group interface consists of the two devices <code>enetA.m1.10-6.0</code> and <code>enetA.m1.11-6.0</code>, the name of the SDLMUX-group interface that you specify for this field is <code>sdlmuxA.m1.10-6.0.11-6.0</code>.</p> |                 |

The STCP protocol stack recognizes the SDLMUX-group interface by the name that you specify for this field.

(For the configuration in [“Sample SDLMUX Configuration” on page 3-7](#), the names of the SDLMUX-group interfaces and ports also indicate the LAN. For information

about creating the device names of ports, see [“Field Descriptions for Ports on Ethernet PCI Adapters” on page 3-14.](#))

- ▶ `module_name` **Required**  
Specify the name of the module that contains the SDLMUX-group interface.
- ▶ `device_type` **Required**  
Specify the type of device that you are configuring. You must specify the value `streams` because an SDLMUX-group interface is a STREAMS device.
- ▶ `access_list_name`  
The name of the device access list to which the SDLMUX-group interface is assigned. For example, you can specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `streams_driver` **Required**  
Specify the value `sdlmux`. This field is required for all devices associated with STCP.
- ▶ `clone_limit` **Required**  
Specify the maximum number of clone devices associated with the SDLMUX-group interface. The appropriate limit is 32; specify the value `32`.

### Entries for SDLMUX-Group Interfaces

The following sample device entries configure the SDLMUX-group interfaces for the sample configuration in [“Sample SDLMUX Configuration” on page 3-7](#) (the configuration procedures are illustrated in [Figure 3-4](#); the device entries for the ports in these SDLMUX-group interfaces appear in [“Entries for Ports on Ethernet PCI Adapters” on page 3-18](#)); examples also exist in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.

```
/ =name          sdlmuxA.m1.10-6.0.11-6.0 /* group on LAN A */
  =module_name    m1
  =device_type     streams
  =access_list_name stcp_access
  =streams_driver  sdlmux
  =clone_limit     32

/ =name          sdlmuxB.m1.10-4.1.11-4.1 /* group on LAN B */
  =module_name    m1
  =device_type     streams
  =access_list_name stcp_access
```

*(Continued on next page)*

```
=streams_driver    sdlmux
=clone_limit       32

/ =name            sdlmuxC.m1.11-3.0          /* group on LAN C */
=module_name       m1
=device_type        streams
=access_list_name  stcp_access
=streams_driver    sdlmux
=clone_limit       32
```

## Creating Device Entries for Ports on Ethernet PCI Adapters

You must create a `devices.tin` file entry for each port on each Ethernet PCI adapter on a module in order to define it as a network interface, associate the port with an SDLMUX-group interface, and if the port is part of a duplexed pair, define its partner. The following sections describe entries for ports on Ethernet PCI adapters.

- [“Field Descriptions for Ports on Ethernet PCI Adapters” on page 3-14](#)
- [“Entries for Ports on Ethernet PCI Adapters” on page 3-18](#)

### Field Descriptions for Ports on Ethernet PCI Adapters

The following `devices.tin` file fields define a port as a network interface, associate it with an SDLMUX-group interface, and if the port is partnered with another port, define its partner.

► **name**

**Required**

Specify the name of the port's network interface. Each device name must be unique, and all names are arbitrary. The name must not exceed 30 characters. By convention, the name should have the prefix `enet`, but each name must be unique beyond the prefix. Stratus recommends that you use information such as the module name, CPU-I/O enclosure number, adapter slot number, and port number to create unique names and to help identify the device. For example, the name `enet.m1.10-3.0` identifies port 0 on an adapter that is located on module `m1` in CPU-I/O enclosure 10, in adapter slot 3; the adapter in slot 3 is an embedded 10/100/1000-Mbps Ethernet PCI adapter (if an adapter has only one port, the port number is 0 by default, and you do not need to specify the number).

The STCP protocol stack and SDLMUX recognize the network interface by the name that you specify for this field.

(For the configuration in [“Sample SDLMUX Configuration” on page 3-7](#), the names of the ports also indicate the LAN.)

► **module\_name**

**Required**

Specify the name of the module that contains the adapter.

- **device\_type** **Required**  
Specify the type of device that you are configuring. Specify the value `streams_pci` for an Ethernet PCI adapter.
- **access\_list\_name**  
The name of the device access list to which the port is assigned. For STCP devices, specify `stcp_access`. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- **streams\_driver** **Required**  
The lower-level device driver that supports the PCI adapter whose port you are configuring. Specify one of the following values; see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) for a list of the PCI adapters that each driver supports.
- `igb` for the IGB driver
  - `ixgbe` for the IXGBE driver
- **hardware\_address** **Required**  
Specify the hardware address of the adapter in the format '`ie/ss/pp`', where `ie` is the hardware ID number of the CPU-I/O enclosure, `ss` is the number of the slot where the adapter is located, and `pp` is the port number. If the adapter has only one port, you do not need to specify a value; the port number is 0, by default.
- [Table 3-1](#) lists values for the hardware address. See the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679) for illustrations of the ports on multiport adapters.

#### NOTE

You **must** enclose within apostrophes ( ' ' ) the arguments and values that you specify in the `hardware_address` field (for example, '`10/6/0`').

**Table 3-1. Values for the hardware\_address Field**

hardware_address 'ie/ss/pp'	Value	Meaning
ie	10	The CPU-I/O enclosure in the topmost position of the cabinet.
	11	The CPU-I/O enclosure directly below CPU-I/O enclosure 10.
ss	n	The slot number of the PCI adapter. To determine the correct slot number, see the section appropriate for your system: <ul style="list-style-type: none"> <li>“<a href="#">PCI-Adapter Slot Numbers of ftServer V 2608, V 4612, V 6616, V 6624, V 6728, V 2810, V 4820, and V 6832 Systems</a>” on page 3-3</li> <li>“<a href="#">PCI-Adapter Slot Numbers of ftServer V 2404, V 4408, V 6408, and V 6512 Systems</a>” on page 3-5</li> </ul>
pp	0	Port 0—For multiport PCI adapters, port A. For embedded PCI adapters, the left port.
	1	Port 1—For multiport PCI adapters, port B. For embedded PCI adapters, the right port.
	2	Port 2—Port C.
	3	Port 3—Port D.

► parameters

Use this field to specify various arguments. You **must** enclose within apostrophes ( ' ' ) the arguments and values that you specify, as in the following example.

```
=parameters '-partner #enetA.m1.11-5.0 -sdlmux #sdlmuxA.m1.10-5.0.11-5.0'
```

The parameters field has the following arguments.

- `-partner #partner_name`—Specify the device name of the network interface that is the partner of the network interface specified in the `name` field. You must begin the name with the pound sign (#) (for example, `-partner #enetA.m1.11-5.0`). Specify this argument only with duplexed adapters.
- `-sdlmux #SDLMUX-group_interface_name`—Specify the name of the SDLMUX-group interface that contains the network interface specified in the `name` field. You must begin the name with the pound sign (#) (for example, `-sdlmux #sdlmuxA.m1.10-5.0.11-5.0`). The `dlmux_admin` command requires the name of the SDLMUX-group interface in the `parameters` field in order to open the associated device of the SDLMUX-group interface and to

create links to its lower-level device drivers. You must specify this argument for a simplex or duplexed adapter.

- `-duplex value`—This argument is deprecated.
- `-speed value`—Specify the speed of the Ethernet communications line to the adapter specified in the `name` field. For `value`, specify a number optionally followed by `M` (for megabits per second or Mbps, the default) or `G` (for gigabits per second or Gbps).

The `igb` and `ixgbe` drivers use `-speed value` to determine the highest transmission speed allowed for auto-negotiation. If a speed is not specified, the driver uses its highest supported transmission speed (1Gbps for `igb`, 10Gbps for `ixgbe`).

- `-mtu value`—Specify the MTU to be used for packets on the interface. For `value`, specify a number from 1500 to 9200. If you specify a number less than 1500 or greater than 9200, `value` will be set to 1500 or 9200, respectively. Do not include the size of the Ethernet header in `value`. Specify the same value as used for the `-mtu` argument of the `ifconfig` command.

Input packets larger than `value` will be discarded. Input flow control thresholds will be adjusted according to `value`.

The default MTU is the largest value the device will support. Typically, the default setting does not need to be changed. In some cases (for example, an adapter that has a small input FIFO), specifying a non-default number for `value` may improve flow control interaction with the Ethernet switch to which the adapter is connected.

- `-min_speed value`—Specify the minimum speed to auto-negotiate. For `value`, specify a number optionally followed by `M` (for megabits per second or Mbps, the default) or `G` (for gigabits per second or Gbps).
- `-rings value`—Specify the number of rings to use. By default, `value` is 4 for 1 Gbps adapters and 16 for 10 Gbps adapters. The minimum value is 1. The specified value is always reduced to the number of CPUs. If `value` is less than 8 for a 10 Gbps connection, performance problems are likely to occur.
- `-no_energy_efficient_ethernet`—Specify to disable the “Energy Efficient Ethernet” option supported by the Intel i350 hardware. You specify this argument if auto-negotiation fails when connected to older hardware.
- `-input_buffers value [/ring]`—Specify the number of 2K-byte input buffers to allocate to receive input. If you specify `/ring`, `value` represents the number of slots per ring; otherwise, `value` represents the total number of slots. By default, `value` is 1024 for 1 Gbps adapters and 10912 for 10 Gbps adapters. The values are adjusted to the range supported by the driver. You

can tune this parameter by consulting the `dot3VosInput...` meters listed in [Table 9-11](#).

- `-output_buffers value [/ring]`—Specify the number of ring slots to allocate for output. If you specify `/ring`, `value` represents the number of slots per ring; otherwise, `value` represents the total number of slots. By default, `value` is 256 per ring for 1 Gbps adapters and 512 per ring for 10 Gbps adapters. The values are adjusted to the range supported by the driver. You can tune this parameter by consulting the `dot3VosOutput...` meters listed in [Table 9-11](#).
- `-use_minimum_vm`—Specify `-no_use_minimum_vm` to allow a particular 10Gbps Ethernet driver to have better performance, if performance is more important than virtual memory. If virtual memory is extremely limited, specify `-use_minimum_vm` on 1Gbps Ethernet drivers to force them to use less memory; however, the amount of virtual memory saved is minimal. These arguments take effect when the Ethernet device is opened (that is, initialized by `dldmux_admin`).

► `clone_limit`

**Required**

Specify the maximum number of clone devices associated with the adapter. The appropriate limit is 32; specify the value `32`.

## Entries for Ports on Ethernet PCI Adapters

The following sample device entries configure the ports on adapters in “[Sample SDLMUX Configuration](#)” on page 3-7 ([Figure 3-4](#) illustrates the configuration procedures); examples also exist in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.

The sample device entries include the `parameters` field with the argument `-sdldmux #SDLMUX-group_interface_name` to specify the name of the SDLMUX-group interface with which the adapter is associated. Sample device entries for duplexed ports also include the argument `-partner #partner_name` to specify the name of the port’s partner in the SDLMUX-group interface.

## NOTE

**Do not enter the plus (+) sign that appears in this example; it is a line-continuation character.** One line in a device entry can extend beyond one terminal-screen line (in this example, the line for the `parameters` field). As you enter characters in a line that extends beyond one terminal-screen line, a plus sign automatically begins the second terminal-screen line to indicate that the line continues on a second terminal-screen line.



**Entries for LAN A**

```

/      =name          enetA.m1.10-6.0
      =module_name    m1
      =device_type    streams_pci
      =access_list_name stcp_access
      =streams_driver  igb
      =hardware_address '10/6/0'
      =parameters     '-partner #enetA.m1.11-6.0
+ -sdlmux #sdlmuxA.m1.10-6.0.11-6.0 -duplex half -speed 100'
      =clone_limit     32

/      =name          enetA.m1.11-6.0
      =module_name    m1
      =device_type    streams_pci
      =access_list_name stcp_access
      =streams_driver  igb
      =hardware_address '11/6/0'
      =parameters     '-partner #enetA.m1.10-6.0
+ -sdlmux #sdlmuxA.m1.10-6.0.11-6.0 -duplex half -speed 100'
      =clone_limit     32

```

**Entries for LAN B**

```

/      =name          enetB.m1.10-4.1
      =module_name    m1
      =device_type    streams_pci
      =access_list_name stcp_access
      =streams_driver  igb
      =hardware_address '10/4/1'
      =parameters     '-partner #enetB.m1.11-4.1
+ -sdlmux #sdlmuxB.m1.10-4.1.11-4.1'
      =clone_limit     32

/      =name          enetB.m1.11-4.1
      =module_name    m1
      =device_type    streams_pci
      =access_list_name stcp_access
      =streams_driver  igb
      =hardware_address '11/4/1'
      =parameters     '-partner #enetB.m1.10-4.1
+ -sdlmux #sdlmuxB.m1.10-4.1.11-4.1'
      =clone_limit     32

```

**Entries for LAN C**

```
/      =name          enetC.m1.11-3.0
      =module_name    m1
      =device_type    streams_pci
      =access_list_name stcp_access
      =streams_driver  igb
      =hardware_address '11/3/0'
      =parameters     '-sdlmux #sdlmuxC.m1.11-3.0'
      =clone_limit     32
```

**Creating, Installing, and Activating the `devices.table` File**

After you define the required SDLMUX-group interfaces and ports in the `devices.tin` file, you must re-create the `devices.table` file, as [“Overview of Device Configuration” on page 3-1](#) describes.

**Verifying Devices**

After you have activated the `devices.table` file, you can confirm the configuration of various devices by using the `list_devices` command. To confirm the device configuration of all STREAMS devices, including SDLMUX-group interfaces and the ports on Ethernet PCI adapters, issue the following command.

```
list_devices -type streams
```

The command output lists all `streams` devices configured on the module; the following example is an excerpt.

```
%es#sdlmuxA.m1.10-6.0.11-6.0
%es#sdlmuxB.m1.10-4.1.11-4.1
%es#sdlmuxC.m1.11-3.0
%es#enetA.m1.10-6.0
%es#enetA.m1.11-6.0
%es#enetB.m1.10-4.1
%es#enetB.m1.11-4.1
%es#enetC.m1.11-3.0
%es#ip.m1
%es#stcp.m10
%es#udp.m1
%es#udp.m1_1
```

To confirm the device configuration of only the ports on the Ethernet PCI adapters, issue the `list_devices` command with the value `streams_pci` for the `-type` argument. The following command is an example.

```
list_devices -type streams_pci
```

If you use the value `pci` for the `-type` argument, the command output includes both ports on the Ethernet PCI adapters and any PCI devices configured on the module. For the configuration in “[Sample SDLMUX Configuration](#)” on page 3-7 (the configuration procedures are illustrated in [Figure 3-4](#)), the command output includes the device names of the SDLMUX-group interfaces `sdlmuxA.m1.10-6.0.11-6.0`, `sdlmuxB.m1.10-4.1.11-4.1`, and `sdlmuxC.m1.11-4.0` as well as the following ports on the Ethernet PCI adapters (assuming that module `m1`, which contains these devices, is part of system `s1`).

```
%s1#enetA.m1.10-6.0
%s1#enetA.m1.11-6.0
%s1#enetB.m1.10-4.1
%s1#enetB.m1.11-4.1
%s1#enetC.m1.11-3.0
```

## Initializing an SDLMUX-Group Interface

You must initialize an SDLMUX-group interface in order to establish it as the logical network interface associated with SDLMUX for the physical network interface (that is, for the ports in the SDLMUX-group interface). You initialize an SDLMUX-group interface by issuing the `dlmux_admin` command for one of the ports in the interface

The following command initializes both the port `device_name` and its partner, if a partner exists as defined in the `devices.tin` file. In the command, `device_name` is the name of one of the ports in an SDLMUX-group interface, and `init_sdlmux` is the value of the `action` argument.

```
dlmux_admin #device_name init_sdlmux
```

### NOTES

1. You issue this command for only **one** port in **each** SDLMUX-group interface.
2. You can initialize an SDLMUX-group interface even if one of the adapters in the interface is broken or not present. In such cases, the `dlmux_admin` command issues the message `WARNING: Device device_name hardware is broken or absent.`

3. The port that you specify has the status `ACTIVE UP` and its partner has the status `UP` (if the command completes with no errors).

The command `dlmux_admin #device_name init_sdldmux` also loads SDLMUX automatically if it is not already loaded. For additional information about loading SDLMUX automatically, see [“Automatically Loading and Unloading SDLMUX” on page 3-23](#).

The `start_stcp.cm` file typically issues the `dlmux_admin` command; thus, you need to customize and activate the `dlmux_admin` command line as it appears in the `start_stcp.cm` file that you receive on the OpenVOS release media.

To customize the `dlmux_admin` command line, replace the device name with the device name of one of the ports in an SDLMUX-group interface in your configuration. To activate the command line, delete the ampersand (&) and space that precede it. If your configuration includes multiple SDLMUX-group interfaces, you **must** add a `dlmux_admin` command line to the `start_stcp.cm` file for **one** port in **each** SDLMUX-group interface.

#### NOTE

The `dlmux_admin #device_name init_sdldmux` command(s) **must** be issued **before** the `ifconfig` command(s) in the `start_stcp.cm` file.

The following `dlmux_admin` commands initialize the ports in [“Sample SDLMUX Configuration” on page 3-7](#). (Figure 3-4 illustrates the configuration procedures; the device entries for these ports appear in [“Entries for Ports on Ethernet PCI Adapters” on page 3-18](#).)

```
dlmux_admin #enetA.m1.10-6.0 init_sdldmux
    (This command also initializes enetA.m1.11-6.0.)
dlmux_admin #enetB.m1.10-4.1 init_sdldmux
    (This command also initializes enetB.m1.11-4.1.)
dlmux_admin #enetC.m1.11-3.0 init_sdldmux
```

To verify the initialization, issue the following command.

```
dlmux_admin #SDLMUX-group_interface_name sdldmux_status
```

The status of the active partner in the SDLMUX-group interface is `ACTIVE UP` when it is initialized; the status of its partner (if one exists) is `UP`.

## Loading and Unloading SDLMUX

SDLMUX (`sdlmux.cp.pm`) must be loaded and unloaded on the module. You can configure SDLMUX to load and unload automatically, or you can load and unload SDLMUX manually. This section describes these procedures as well as how to verify that SDLMUX is loaded.

- [“Automatically Loading and Unloading SDLMUX” on page 3-23](#)
- [“Manually Loading and Unloading SDLMUX” on page 3-23](#)
- [“Verifying SDLMUX” on page 3-25](#)

### Automatically Loading and Unloading SDLMUX

If SDLMUX is not already loaded, STREAMS automatically loads SDLMUX when you initialize the first port. The following command initializes both the port `device_name` and its partner (as defined in the `devices.tin` file); the command also loads SDLMUX if it is not already loaded.

```
dlmux_admin #device_name init_sdlmux
```

If SDLMUX loads automatically, it unloads automatically when the last active partnered pair of ports is uninitialized. The following command uninitializes both the port `device_name` and its partner (as defined in the `devices.tin` file); the command also unloads SDLMUX (if it was loaded automatically).

```
dlmux_admin #device_name uninit_sdlmux
```

The `start_stcp.cm` file typically issues the `dlmux_admin` command; thus, you need to customize and activate the `dlmux_admin` command line as it appears in the `start_stcp.cm` file that you receive on the OpenVOS release media.

For additional information about initializing partnered ports, see [“Initializing an SDLMUX-Group Interface” on page 3-21](#).

### Manually Loading and Unloading SDLMUX

You can load SDLMUX manually by using either the command `configure_comm_protocol` or the command `load_kernel_program`. If you load SDLMUX manually, you must still initialize the ports by using the command `dlmux_admin #device_name init_sdlmux`.

The command `configure_comm_protocol` searches for SDLMUX in the `(master_disk)>system>kernel_loadable_library` directory and loads it if it is not already loaded. If SDLMUX is already loaded, the command

`configure_comm_protocol` does nothing. To use `configure_comm_protocol`, issue the following command.

```
configure_comm_protocol sdlmux
```

The command `load_kernel_program` searches for SDLMUX using the path name specified in the command line. If SDLMUX is already loaded, the `load_kernel_program` command reports an error. To use `load_kernel_program`, issue the following command.

```
load_kernel_program (master_disk)>system>kernel_loadable_library>sdlmux.cp
```

Or, alternatively, change your current directory to the directory `(master_disk)>system>kernel_loadable_library`, and issue the following command.

```
load_kernel_program sdlmux.cp
```

If you load SDLMUX manually, you must unload it manually by using either the command `delete_comm_protocol` or the command `unload_kernel_program`. To use `delete_comm_protocol`, issue the following command.

```
delete_comm_protocol sdlmux
```

To use `unload_kernel_program`, issue the following command.

```
unload_kernel_program  
(master_disk)>system>kernel_loadable_library>sdlmux.cp
```

Or, alternatively, change your current directory to the directory `(master_disk)>system>kernel_loadable_library` and issue the following command.

```
unload_kernel_program sdlmux.cp
```

For descriptions of the commands `configure_comm_protocol` and `delete_comm_protocol`, see the manual *OpenVOS System Administration: Configuring a System* (R287). For descriptions of the commands `load_kernel_program` and `unload_kernel_program`, see the manual *OpenVOS System Administration: Administering and Customizing a System* (R281).

## Verifying SDLMUX

To verify that SDLMUX is loaded, issue either the command `list_comm_protocols` or the command `list_kernel_programs`. When SDLMUX is loaded, the output of either command includes a line similar to the following.

```
%s1#m1>system>kernel_loadable_library>sdlmux.cp.pm
```

The output also includes a line similar to the following when the lower-level device driver is loaded, where *driver* is *igb* or *ixgbe*.

```
%s1#m1>system>kernel_loadable_library>driver.cp.pm
```

The lower-level device driver (*igb*, or *ixgbe*) is loaded automatically by STREAMS; you do not need to explicitly load it. These drivers are located in the `(master_disk)>system>kernel_loadable_library` directory.

For a description of the `list_comm_protocols` command, see the manual *OpenVOS System Administration: Configuring a System* (R287). For a description of the `list_kernel_programs` command, see the manual *OpenVOS System Administration: Administering and Customizing a System* (R281).

## Defining the SDLMUX-Group Interfaces

You must define the SDLMUX-group interfaces to the STCP protocol stack by using the `ifconfig` command. The `start_stcp.cm` file typically contains `ifconfig` command lines to configure these interface. Thus, you must modify existing and/or add new `ifconfig` commands to the `start_stcp.cm` file. Command lines similar to the following configure an interface for IPv4 support.

```
ifconfig #SDLMUX-group_interface_name name_or_address
-netmask string -add
```

For each interface configured for IPv6 support, use classless inter-domain routing (CIDR) notation for the `name_or_address` argument. If you include `/N` when specifying `name` (of the `name_or_address` argument), you cannot specify the `-netmask` argument.

The IP address that you specify applies to the SDLMUX-group interface. Thus, the SDLMUX-group interface serves as the logical representation of the port(s) in the interface, and the ports are the physical network interfaces.

### NOTE

The `ifconfig` command(s) **must** be issued **after** the `dlnux_admin #device_name init_sdlmux` command(s). Since the `start_stcp.cm` file typically

issues both the `dlmux_admin #device_name`  
`init_sdlmux` command(s) and the `ifconfig`  
command(s), the `ifconfig` command(s) must appear in  
the `start_stcp.cm` file **after** the `dlmux_admin`  
`#device_name init_sdlmux` command(s).

The following `ifconfig` command lines define the SDLMUX-group interfaces in “[Sample SDLMUX Configuration](#)” on page 3-7. (Figure 3-4 illustrates the configuration procedures; the device entries for these devices appear in “[Entries for SDLMUX-Group Interfaces](#)” on page 3-13.)

```
ifconfig #sdlmuxA.m1.10-6.0.11-6.0 172.16.2.10 -netmask 0xffffffff00 -add
ifconfig #sdlmuxB.m1.10-4.1.11-4.1 172.16.3.10 -netmask 0xffffffff00 -add
ifconfig #sdlmuxC.m1.11-3.0 172.16.4.10 -netmask 0xffffffff00 -add
```

The following example includes CIDR notation for an interface configured for IPv6 support:

```
ifconfig #sdlmuxC.m1.11-3.0 172.16.4.10 2001:1900:3008:fc85::64/64 -add -alias
```

To verify the definition of these SDLMUX-group interfaces, issue one of the following commands.

- **`ifconfig`**—This command (with no arguments) displays the status of all STCP interfaces that are configured on the module.
- **`netstat -interface`**—You can issue the STCP command `netstat -interface` to display information about SDLMUX-group interfaces after you have defined the devices using the `ifconfig` command. When you issue the command for the SDLMUX-group interface, the command output indicates that the network interfaces are grouped as well as the number of failovers that have occurred.

## Changing Existing Configurations

This section describes procedures for the following configuration changes.

- “[Deleting an SDLMUX-Group Interface and Its Ports](#)” on page 3-27
- “[Deleting or Adding a Partner Port in an SDLMUX-Group Interface](#)” on page 3-27
- “[Adding a Simplex Port](#)” on page 3-30



## Deleting an SDLMUX-Group Interface and Its Ports

The following steps delete an SDLMUX-group interface and its ports from a configured network.

1. Delete the SDLMUX-group interface from the STCP protocol stack. If users are connected through the ports in the SDLMUX-group interface, ask them to log off; deleting the SDLMUX-group interface will disconnect the processes that use it. Issue the following `ifconfig` command to delete the SDLMUX-group interface.

```
ifconfig #SDLMUX-group_interface_name -delete
```

To permanently remove this SDLMUX-group interface from your configuration, you must also delete from the `start_stcp.cm` file the `ifconfig` command line that defines (that is, adds) this SDLMUX-group interface to the STCP protocol stack.

2. Issue the following `dlmux_admin` command to uninitialized both the port `device_name` and its partner, if one exists, as defined in the `devices.tin` file.

```
dlmux_admin #device_name uninit_sdlnmux
```

To permanently uninitialized the ports from your configuration, you must also delete the `dlmux_admin #device_name init_sdlnmux` command line for the port `device_name` from the `start_stcp.cm` file.

(If appropriate, physically remove the adapter of the port that you are deleting.)

3. To permanently delete the SDLMUX-group interface and the ports in the interface, you must also modify the `devices.tin` file, deleting the entries for the SDLMUX-group interfaces and for the individual ports that you removed. Then, re-create, install, and immediately activate the `devices.table` file by issuing the following commands.

```
create_table devices
broadcast_file devices.table >system
configure_devices -flush
```

## Deleting or Adding a Partner Port in an SDLMUX-Group Interface

This section describes how to delete a partner port from or add a partner port to an existing, configured SDLMUX-group interface. This section discusses the following topics.

- [“Deleting a Partner Port” on page 3-28](#)
- [“Adding a Partner Port” on page 3-29](#)

## Deleting a Partner Port

This procedure deletes one partner port from an existing, configured SDLMUX-group interface.

1. If the port that you are deleting is the active partner, you need to perform a manual hardware failover. Issue the following command in order to make the port *device\_name* inactive and to perform the manual hardware failover.

```
dlmux_admin #device_name failover
```

2. Delete the inactive port by issuing the following command, where *device\_name* is the name of the **active** partner.

```
dlmux_admin #device_name delete_partner
```

You cannot delete an active partner (that is, a partner that is currently handling communications I/O); attempting to delete an active partner generates an error.

3. In the `devices.tin` file, make the following changes:
  - Change the entry for the port that remains in the existing SDLMUX-group interface. Delete the argument `-partner #partner_name` in the `parameters` field.
  - Delete the entry for the port that you are deleting from the SDLMUX-group interface.

Then, re-create, install, and immediately activate the `devices.table` file by issuing the following commands.

```
create_table devices
broadcast_file devices.table >system
configure_devices -flush
```

4. Check the `start_stcp.cm` file for the `dlmux_admin` command line that initializes these ports (`dlmux_admin #device_name init_sdlmux`). If the `start_stcp.cm` file initializes the port that you have deleted from the existing SDLMUX-group interface, the `start_stcp.cm` file must now initialize the port that remains in the existing SDLMUX-group interface. Change the `dlmux_admin #device_name init_sdlmux` command line, specifying the name of the port that remains in the existing SDLMUX-group interface. Save the modified `start_stcp.cm` file.

To immediately initialize the port that was not initialized, issue the `dlmux_admin #device_name init_sdlmux` command with the name of the appropriate port.

## Adding a Partner Port

To add a partner port (during the current bootload) to an existing, configured SDLMUX-group interface that contains only one port, perform the following steps.

1. Install the new adapter in the correct slot in the appropriate CPU-I/O enclosure. Connect the port on the new adapter to the same LAN as the existing port.
2. Check that the existing port is correctly configured.
  - a. Check that the `start_step.cm` file contains the following commands.
    - the command `dlnmux_admin #device_name init_sdlnmux` to initialize the existing ports
    - the command `ifconfig` to add the associated SDLMUX-group interface
  - b. In the `devices.tin` file entry for the existing port, check that the parameters have the correct values. You need to add the argument `-partner #partner_name` to specify the name of the existing port's new partner in the SDLMUX-group interface.
3. Create a `devices.tin` file entry for the new port, as described in “[Creating Device Entries for Ports on Ethernet PCI Adapters](#)” on page 3-14. Be sure to specify a correct value for the `-partner #partner_name` argument.

Then, create, install, and immediately activate the `devices.table` file by issuing the following commands.

```
create_table devices
broadcast_file devices.table >system
configure_devices
```

4. Issue the following command to associate the existing port (specified in the `device_name` argument) with the new partnered port (specified in the argument `-partner new_partner_name`). (See the `dlnmux_admin` command description in Chapter 4 for more information.)

```
dlnmux_admin #device_name add_partner -partner new_partner_name
```

You can add a port to an SDLMUX-group interface even if the adapter is broken or not present. The port must have a `devices.tin` file entry. In such cases, the `dlnmux_admin` command issues the message `WARNING: Device device_name hardware is broken or absent`.

If you replace one type of Ethernet adapter with a different type of Ethernet adapter, you do **not** receive a warning that the adapter is of a different type, even if the network experiences problems.

## Adding a Simplexed Port

To add a simplexed port to a configuration, perform the following steps.

1. Physically insert the new adapter that has the port that you are adding.
2. In the `devices.tin` file, make the following changes, as described in [“Creating Device Entries for Ports on Ethernet PCI Adapters” on page 3-14](#).
  - Create an entry for the new simplexed port, including the `parameters` field with correct values for the `-sdlmux SDLMUX-group_interface_name` argument.
  - Create an entry for a new SDLMUX-group interface, which contains the simplexed port.

Then, create, install, and immediately activate the `devices.table` file by issuing the following commands.

```
create_table devices
broadcast_file devices.table >system
configure_devices
```

3. Initialize the SDLMUX-group interface by issuing the following command for the simplexed port.

```
dlmux_admin #device_name init_sdlmux
```

To permanently add this simplexed port to your configuration, you must also add this command line to the `module_start_up.cm` file.

4. Define the new SDLMUX-group interface to the STCP protocol stack by issuing the following command.

```
ifconfig #SDLMUX-group_interface_name name_or_address
-netmask string -add
```

To permanently add this SDLMUX-group interface to your configuration, you must also add the new `ifconfig` command line for the SDLMUX-group interface to the `start_stcp.cm` file.

## SDLMUX Requests of the *analyze\_system* Subsystem

The *analyze\_system* subsystem includes the *dump\_sdlmux* request. This request displays information about the structures associated with an SDLMUX-group interface that has been initialized. Specifically, the request displays the names of the two network interface cards in one initialized SDLMUX-group interface.

For more information about this and other requests of the *analyze\_system* subsystem, see the *OpenVOS System Analysis Manual* (R073).



---

## Chapter 4

# Configuring STCP Devices

For an STCP configuration, the `devices.tin` file must include entries for the following devices.

- the protocol drivers in the STCP stack (clonable STREAMS drivers), namely STCP, UDP, loopback, and IP
- the two TELNET pipe devices (one for the TELNET daemon, `telnetd`, and one for the `telnet_admin` command)
- the transport-layer interface (TLI) access-layer device driver, a STREAMS driver that is part of the TLI software and handles STCP TELNET connections
- TELNET login devices as well as TELNET incoming and outgoing slave devices

On systems that run applications requiring POSIX-compliant UNIX<sup>®</sup>-domain sockets, the STCP configuration must include the AF UNIX (`af_unix`) driver. For these systems, the `devices.tin` file must include an entry for the `af_unix` driver.

Entries for the required STCP devices and for the `af_unix` driver exist in the sample `devices.tin` file, which is the file `sample_devices.release_no` in the `(master_disk)>system>configuration` directory. If necessary, you can modify these entries. If your configuration requires additional entries, you can copy existing entries and modify them, as necessary.

In this chapter, the following sections describe STCP device entries. Most entries contain required fields. The order of fields in the descriptions matches the order of fields in the `devices.dd` file. This order is not mandatory; you can rearrange the order of the fields when you create an entry.

- [“Defining STCP Protocol Drivers” on page 4-2](#)
- [“Defining STCP TELNET Devices” on page 4-8](#)

The remaining sections in this chapter provide additional information about configuring STCP devices.

- [“Creating, Installing, and Activating the `devices.table` File” on page 4-30](#)
- [“Controlling Access to STCP Devices” on page 4-30](#)

For an overview of creating entries in this file, see [“Overview of Device Configuration” on page 3-1](#). An STCP configuration must also include entries for NICs and their SDLMUX-group interfaces. If these entries do not exist, you must create them (see [Chapter 3](#)). Many systems use pseudo-terminals as part of the STCP configuration. Pseudo-terminals provide support for terminal devices that connect to a process rather than to hardware. For information on pseudo-terminals, see *OpenVOS System Administration: Configuring a System* (R287).

## Defining STCP Protocol Drivers

You must create device entries for the STCP protocol drivers (clonable STREAMS drivers) in the `devices.tin` file if these entries do not already exist. The following sections explain how to create the device entries for the STCP protocol drivers.

- [“Defining the STCP Driver” on page 4-2](#)
- [“Defining the UDP Driver” on page 4-4](#)
- [“Defining the IP Driver” on page 4-5](#)
- [“Defining the Loopback Driver” on page 4-6](#)

### Defining the STCP Driver

The `devices.tin` file must contain an entry for the STCP driver. This section provides sample entries and descriptions of the fields required to define the driver. A sample entry also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory. (For information about controlling access to the STCP driver, see [“Controlling Access to STCP Devices” on page 4-30](#).)

- The following entry is for a module that is running OpenVOS Release 17.2 (or later) and requires a very large number (240,000) of sockets:

```
/      =name                stcp.m1
      =module_name          m1
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       stcp
      =clone_limit          16000
      =clone_limit_32       240000
```



- The following entry is for any module that requires a large number (15,000) of sockets:

```

/      =name          stcp.m1
      =module_name    m1
      =device_type    streams
      =access_list_name stcp_access
      =streams_driver  stcp
      =clone_limit     15000

```

- The following entry is for any module that requires a typical number (5,120) of sockets:

```

/      =name          stcp.m1
      =module_name    m1
      =device_type    streams
      =access_list_name stcp_access
      =streams_driver  stcp
      =clone_limit     5120

```

Specify information in the fields as follows:

- ▶ `name`  
Specify the name of the STCP driver on the module. The name must have the prefix `stcp.`, followed by the name of the module (for example, `stcp.m1`).
- ▶ `module_name`  
Specify the name of the module on which you are configuring the STCP driver (for example, `m1`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.
- ▶ `access_list_name`  
The name of the device access control list (ACL) file to which the STCP driver is assigned. For example, you could specify `stcp_access` as the name of the device ACL file for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device ACL file. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `streams_driver`  
Specify the name of the STCP driver. You must specify the value `stcp`.
- ▶ `clone_limit`  
Specify the clonable device limit for the STCP driver. On `ftServer` modules, specify the value `5120`, which is a suitable value for handling up to approximately 5,120

connections between applications and STCP. The maximum value on modules running OpenVOS Release 17.2 (or greater) is 16000. However, if you specify a value greater than 5120 on any module, you increase the risk of runaway applications or denial-of-service attacks using extreme amounts of system resources. If you want a limit larger than 16,000 on modules running OpenVOS Release 17.2 (or greater), specify the larger limit in the `clone_limit_32` field.

► `clone_limit_32`

Specify the clonable device limit for the STCP driver on modules running OpenVOS 17.2 or later. Values range from the value specified in `clone_limit` to the maximum number of devices that the module supports. However, if you specify a value greater than 5120 on any module, you increase the risk of runaway applications or denial-of-service attacks using extreme amounts of system resources. If you specify a value for `clone_limit_32` you should also specify a value for `clone_limit`. The value of `clone_limit` should be the minimum of 16,000 and the `clone_limit_32` value.

## Defining the UDP Driver

The `devices.tin` file must contain an entry for the UDP driver. The following sample entry shows the fields required to define this driver; an example also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.

```
/      =name                udp.m1
      =module_name         m1
      =device_type         streams
      =access_list_name    stcp_access
      =streams_driver      udp
      =clone_limit         5120
```

Specify information in the fields as follows:

► `name`

Specify the name of the UDP driver on the module. The name must have the prefix `udp.`, followed by the name of the module (for example, `udp.m1`).

► `module_name`

Specify the name of the module on which you are configuring the UDP driver (for example, `m1`).

► `device_type`

Specify the type of device that you are configuring. You must specify the value `streams`.

- ▶ `access_list_name`  
The name of the device access list to which the UDP driver is assigned. For example, you could specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `streams_driver`  
Specify the name of the STREAMS driver that you are configuring. You must specify the value `udp`.
- ▶ `clone_limit`  
Specify the clonable device limit for the driver that you are configuring. Specify the value `5120`, which is a suitable value for handling up to approximately 5000 connections between applications and UDP. This is the maximum number of connections allowed.

## Defining the IP Driver

The `devices.tin` file must contain an entry for the IP driver. The following sample entry shows the fields required to define this driver; an example also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.

```
/    =name           ip.m1
    =module_name     m1
    =device_type      streams
    =access_list_name stcp_access
    =streams_driver   ip
    =clone_limit      1024
```

Specify information in the fields as follows:

- ▶ `name`  
Specify the name of the IP driver on the module. The name must have the prefix `ip.`, followed by the name of the module (for example, `ip.m1`).
- ▶ `module_name`  
Specify the name of the module on which you are configuring the IP driver (for example, `m1`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.

- ▶ `streams_driver`  
Specify the name of the STREAMS driver that you are configuring. You must specify the value `ip`.
- ▶ `access_list_name`  
The name of the device access list to which the IP driver is assigned. For example, you could specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `clone_limit`  
Specify the clonable device limit for the driver that you are configuring. Specify the value `1024`. This is the maximum number of connections allowed.

## Defining the Loopback Driver

You must create a device entry in the `devices.tin` file for the loopback driver. The following sample entry shows the fields required to define this driver; an example also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory.

```
/    =name                loop.m1
    =module_name          m1
    =device_type          streams
    =access_list_name     stcp_access
    =streams_driver       loop
    =clone_limit          1024
```

Specify information in the fields as follows:

- ▶ `name`  
Specify the name of the loopback driver on the module. The name must have the prefix `loop.`, followed by the name of the module (for example, `loop.m1`).
- ▶ `module_name`  
Specify the name of the module on which you are configuring the loopback driver (for example, `m1`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.
- ▶ `access_list_name`  
The name of the device access list to which the loopback driver is assigned. For example, you could specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the

give\_access command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))

- ▶ `streams_driver`  
Specify the name of the STREAMS driver that you are configuring. You must specify the value `loop`.
- ▶ `clone_limit`  
Specify the clonable device limit for the driver that you are configuring. Specify the value `1024`. This is the maximum number of connections allowed.

## Defining the AF UNIX Driver

If your configuration uses POSIX-compliant UNIX-domain sockets, you must create an entry in the `devices.tin` file for the AF UNIX driver, if the entry does not already exist. The following sample entry shows the fields required to define this driver; an example also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory. (For information about controlling access to the STCP driver, see [“Controlling Access to STCP Devices” on page 4-30.](#))

```
/    =name                s$af_unix.ml
    =module_name         ml
    =device_type          streams
    =access_list_name     stcp_access
    =streams_driver       af_unix
    =clone_limit          4096
```

Specify information in the fields as follows:

- ▶ `name`  
Specify the name of the AF UNIX driver on the module. The name must be `s$af_unix.`, followed by the name of the module (for example, `s$af_unix.ml`).
- ▶ `module_name`  
Specify the name of the module on which you are configuring the AF UNIX driver (for example, `ml`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.
- ▶ `access_list_name`  
The name of the device access list to which the AF UNIX driver is assigned. For example, you could specify `stcp_access`. After you have installed the

`devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see “[Controlling Access to STCP Devices](#)” on page 4-30.)

- ▶ `streams_driver`  
Specify the name of the STREAMS driver that you are configuring. You must specify the value `af_unix`.
- ▶ `clone_limit`  
Specify the clonable device limit for the driver that you are configuring. Specify the value `4096`. This is the maximum number of connections allowed.

For information on POSIX compliance and OpenVOS, see the following documentation:

- *OpenVOS POSIX.1: Conformance Guide* (R217M), which describes how the OpenVOS POSIX.1 implementation adheres to or deviates from the POSIX standard. This document is available only on the OpenVOS StrataDOC Web site.
- *OpenVOS POSIX.1 Reference Guide* (R502), which documents OpenVOS POSIX features.

## Defining STCP TELNET Devices

If you want the module to handle TELNET connections by using the STCP TELNET daemon process (`telnetd`), you must define a number of TELNET device entries in the `devices.tin` file, as follows:

- You must create one device entry to define the TLI access-layer device driver (`tli_al`). This entry identifies the driver's governing STREAMS driver, `tli_term`. The TLI access-layer device driver communicates with the upper layers of the window terminal software and works with the `telnetd` daemon process to handle the module's TELNET connections.
- You must create two device entries that define the STCP TELNET pipe devices, one for the `telnetd` daemon and one for the `telnet_admin` command.
- You must create a clonable device entry (or individual device entries joined by a common prefix) for each type of STCP TELNET *login device*, which represents a local TELNET login service and handles a TELNET user's connection to that service. A login device enables remote users on the network to use TELNET to log in to the module. Each type of login device that you define in a device entry accommodates a remote user's incoming connection request to the module. (There is a default login TELNET service that uses port 23; if you need to define an additional login service that does not use the same options as the default service, you can define the service information in the STCP `telnet-service` database file.)

- If your configuration includes local slave services, you can create a clonable device entry (or individual device entries joined by a common prefix) for each type of STCP TELNET *incoming slave device*, which represents a local slave service and handles a client-initiated slave connection to that service. An incoming slave device enables a remote user to use TELNET to connect to an available local service (application) running on the module. The incoming slave device that you define in a device entry is associated with a service on the module; to connect to the service, the slave device must already be attached and opened by an active application on the module. (The service for an incoming slave device must also be defined in the STCP `telnet-service` database file as well as the `services` database file.) In essence, an application opens a port to the device and waits for a remote user to initiate a connection request to the device's service.
- If your configuration includes network slave services, you can create a clonable device entry (or individual device entries joined by a common prefix) for each type of STCP TELNET *outgoing slave device*, which represents a network service and handles a host-initiated (OpenVOS-initiated) slave connection to that service. An outgoing slave device establishes an OpenVOS-initiated connection between an application running on the STCP module and a remote destination on the network. Each outgoing slave device that you define in a device entry is associated with a network service. The application running on the module attaches and opens a port to the outgoing slave device; the access layer then initiates the remote connection on behalf of the application. For example, you may configure an outgoing slave device that represents a remote network printer. If the OpenVOS spooler process running on the module attaches and opens a port to this outgoing slave device, the access layer can request a connection to the network printer in a manner that is transparent to the OpenVOS spooler process.

The following sections describe how to create the different types of entries for STCP TELNET devices.

- [“Defining the TLI Access-Layer Device Driver” on page 4-9](#)
- [“Defining STCP TELNET Pipe Entries” on page 4-10](#)
- [“Defining STCP TELNET Login Devices” on page 4-12](#)
- [“Defining STCP TELNET Incoming Slave Devices” on page 4-20](#)
- [“Defining STCP TELNET Outgoing Slave Devices” on page 4-26](#)

## Defining the TLI Access-Layer Device Driver

You must create a device entry for the TLI access-layer device driver, which is governed by the STREAMS driver `tli_term`.

The following sample device entry defines the TLI access-layer device driver; an example also exists in the `sample_devices.release_no` file in the `(master_disk)>system>configuration` directory. In the entry, the TLI

access-layer device driver on #m1 is assigned the name `tli_al.m1` and uses the `tli_term` STREAMS driver.

```
/      =name                tli_al.m1
      =module_name          m1
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       tli_term
      =minor_number         0
```

Specify information in the fields as follows:

- ▶ `name`  
Specify a unique name for the TLI access-layer device driver on the module. The name **must** have the prefix `tli_al.`, followed by the name of the module (for example, `tli_al.m1`).
- ▶ `module_name`  
Specify the name of the module on which you are configuring the access-layer device driver (for example, `m1`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.
- ▶ `access_list_name`  
The name of the device access list to which the TLI access-layer device driver is assigned. For example, you could specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `streams_driver`  
Specify the type of STREAMS device driver that you are configuring. You must specify the value `tli_term`.
- ▶ `minor_number`  
Specify the value 0; do not change this value.

## Defining STCP TELNET Pipe Entries

You must create two TELNET pipe entries, one to accommodate a pipe for `telnetd` and one to accommodate a pipe for the `telnet_admin` command.

The following sample entries define the TELNET pipe entry for the daemon process and the entry for the `telnet_admin` command; examples also exist in the



`sample_devices.release_no` file in the  
(master\_disk)>system>configuration directory. Both entries rely on a `tpipe`  
driver that STREAMS automatically loads on the module.

```

/      =name                tpipe_telnetd.m1
      =module_name          m1
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       tpipe
      =minor_number        0

/      =name                tpipe_admin.m1
      =module_name          m1
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       tpipe
      =minor_number        1

```

Specify information in the fields as follows:

- ▶ `name`  
Specify the correct name for each pipe device, as follows:
  - For the name of the `telnetd` daemon pipe, specify the prefix `tpipe_telnetd.`, followed by the module name (for example, `tpipe_telnetd.m1`).
  - For the name of the `telnet_admin` pipe, specify the prefix `tpipe_admin.`, followed by the module name (for example, `tpipe_admin.m1`).
- ▶ `module_name`  
Specify the name of the module that provides the pipe device (for example, `m1`).
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `streams`.
- ▶ `access_list_name`  
The name of the device access list to which the pipe device is assigned. For example, you could specify `stcp_access` for STCP devices. After you have installed the `devices.table` file and started the module, issue the `give_access` command to set access on the device access list. (For more information about access control, see [“Controlling Access to STCP Devices” on page 4-30.](#))
- ▶ `streams_driver`  
Specify the type of STREAMS device driver that you are configuring. You must specify the value `tpipe`.

► `minor_number`

Specify the value 0 for the `tpipe_telnetd` device; specify 1 for the `tpipe_admin` device.

## Defining STCP TELNET Login Devices

To enable the `telnetd` daemon process to handle login requests from TELNET users, you must perform the following steps.

1. Determine whether the default TELNET login service (as defined in the template `telnetSERVICE` database file) is the only type of login service you need to provide. The default TELNET login service permits both privileged and nonprivileged logins. For example, you may want to do the following:
  - You may want to control privileged (and nonprivileged) logins separately, for security reasons, and use a source IP address and a range of ports to control access. You may want to maintain the default login service as privileged and simply add a login service that is only nonprivileged. Or, you may want to change the default login service to nonprivileged and add a login service that handles only privileged logins based on a source IP address and a specific range of ports.
  - You may want to create a login service that employs the `linger` option (and/or varies the setting of the `keepalive` and `no-delay` options). (See “[The telnetSERVICE File and TELNET Information](#)” on page 5-53 for a description of these options.)
2. Select a method for creating the STCP TELNET login device entries. Select one of two methods, as follows:
  - Create a single clonable device entry for each type of login service. In this case, the entire device name of the clonable device is used as the device prefix for the service in the `telnetSERVICE` database file (or the device prefix specified with the `telnet_admin` command if you are adding a service after the stack is up). The `clone limit` enables `telnetd` to automatically create login devices for a service when it needs them.
  - As an alternative, create all of the device entries for a particular login service individually and use a common prefix for the device name to group the entries (that is, use a common prefix and different suffixes). You would then specify the device prefix for the service in the `telnetSERVICE` database file (or with `telnet_admin`).
3. Make sure that the `telnetSERVICE` database file contains the correct information for each login service based on your device entry (or entries) for each service, and make sure that the `SERVICES` database file contains the appropriate service name and port number for the service (the default TELNET login service is defined for port 23). To create (or modify) a service, you either edit the `telnetSERVICE` and

services database files or issue the `telnet_admin` command, which updates both files and is used when the stack is already running on the module.

Figure 4-1 shows two TELNET login services with clonable device entries in the `devices.tin` file, two entries in the `telnet-service` database file, and two entries in the `services` database file. Important information is shown in boldface. The following list summarizes the information in the entries.

- The first clonable device entry accommodates a privileged login service that represents the default TELNET login service.
- The second clonable device entry accommodates a nonprivileged login service.
- The entire device name specified in the device entry for each service matches the device prefix specified for each service in the `telnet-service` database file (the last field).
- The values specified in the `login_slave` and `priv_terminal` fields of the device entry for each service match the values specified for the `login_slave` and `privileged` arguments in the `telnet-service` database file.
- The service name specified for each login service in the `telnet-service` database file (the first field) matches the service name specified for each service in the `services` database file.

If you specify a clonable device entry for a service, issuing the `list_devices` command with the `-type window_term` argument will list the cloned TLI login devices (for example, `tli_logm1_1` and `tli_logm1_2`.) The Transport Layer Interface (TLI), which is built into the OpenVOS kernel, works with the window terminal software to support STCP TELNET connections.

**Figure 4-1. Login Device Information with Clonable Device Entries**

**devices.tin File Information**

```
/* clonable device entry for the default login service */
/  =name          tli_logm1 /* matches last field of telnet-service */
    =module_name   m1
    =terminal_type  ascii
    =device_type   window_term
    =login_slave   1
    =priv_terminal  1
    =dialup        0
    =clone_limit   1000

/* clonable device entry for a nonpriv login service */
/  =name          tli_nplogm1
    =module_name   m1
    =terminal_type  ascii
```

*(Continued on next page)*

```
=device_type      window_term
=login_slave      1
=priv_terminal    0
=dialup           0
=clone_limit      1000
```

**telnet service File Information**

```
telnet window_term "keepalive nodelay" "Default Telnet login" 1 1 tli_logm1
mlnplog window_term "keepalive nodelay" "Nopriv login" 1 0 tli_nplogm1
```

**services File Information**

```
telnet      23/tcp      telnetd    # alias
mlnplog     955/tcp
```

Figure 4-2 shows one TELNET login service with individual entries instead of clonable device entries in the `devices.tin` file, an entry in the `telnet service` database file, and an entry in the `services` database file. Important information is shown in boldface. The following list summarizes the information in the entries.

- The device entries use the same prefix (**tli\_logm1**) but different suffixes to define login devices for the default TELNET login service.
- The device prefix specified in both device entries matches the device prefix specified for the service in the `telnet service` database file (the last field).
- The values specified in the `login_slave` and `priv_terminal` fields of both device entries for the service match the values specified for the `login_slave` and `privileged` arguments in the `telnet service` database file.
- The service name specified for the login service in the `telnet service` database file (the first field) matches the service name specified for the service in the `services` database file.

**Figure 4-2. Login Device Information with Individual Device Entries**

**devices.tin File Information**

```
/* First device for the default login service */
/  =name          tli_logm1.1 /* just prefix appears in telnet service*/
   =module_name    m1
   =terminal_type  ascii
   =device_type    window_term
   =login_slave    1
   =priv_terminal  1
   =dialup         0
   =parameters    '-access_layer tli_al'
```

*(Continued on next page)*

```

/* Second device for the default login service */
/ =name          tli_logm1.2
  =module_name    m1
  =terminal_type  ascii
  =device_type    window_term
  =login_slave    1
  =priv_terminal  1
  =dialup         0
  =parameters    '-access_layer tli_al'

telnetSERVICE File Information
telnet window_term "keepalive nodelay" "Default Telnet login" 1 1 tli_logm1

services File Information
telnet          23/tcp          telnetd      # alias

```

Table 4-1 describes the relationship between the fields in the device entry for a login service and the fields in the `telnetSERVICE` file (and with the command arguments that can be issued with the `telnet_admin` command).

**Table 4-1. Key Values for Defining Login Devices** (Page 1 of 2)

devices.tin Field	telnetSERVICE File Argument or telnet_admin Argument	Explanation
name	<code>device_prefix</code> or <code>-device_prefix</code>	In a clonable device entry for a login service, the entire name that you specify in the <code>name</code> field must match the name that you specify for the <code>device_prefix</code> argument in the <code>telnetSERVICE</code> file (or for the <code>-device_prefix</code> argument of the <code>telnet_admin</code> command). If you specify individual device entries for a login service, each entry for the service must have the same prefix and different suffixes in the <code>name</code> field, and the prefix must match the prefix that you specify for the <code>device_prefix</code> argument in the <code>telnetSERVICE</code> file (or for the <code>-device_prefix</code> argument of the <code>telnet_admin</code> command). The name or prefix for the default login service should be <code>tli_logmx</code> , where <code>x</code> identifies the module. For ease of reference, you may want the names of all STCP TELNET services to begin with <code>tli</code> .
login_slave	<code>login_slave</code> or <code>-login</code>	For each login device associated with a login service, specify the value 1 in the <code>login_slave</code> field and specify the value 1 for the <code>login_slave</code> argument in the <code>telnetSERVICE</code> file (or specify the <code>-login</code> argument of the <code>telnet_admin</code> command).

**Table 4-1. Key Values for Defining Login Devices** (Page 2 of 2)

<code>devices.tin</code> Field	telnet service File Argument or telnet_admin Argument	Explanation
<code>priv_terminal</code>	<i>privileged</i> or <i>-privileged</i>	If you want the login devices associated with a particular login service to support <b>both</b> nonprivileged and privileged users, specify the value 1 in the <code>priv_terminal</code> field and specify a 1 for the <i>privileged</i> argument of the telnet service file (or specify the <i>-privileged</i> argument of the <code>telnet_admin</code> command). If you want the login devices associated with a login service to support nonprivileged users only, specify a 0 in the <code>priv_terminal</code> field and specify a 0 for the <i>privileged</i> argument of the telnet service file (or specify the <i>-no_privileged</i> argument of the <code>telnet_admin</code> command). If necessary, you can create two services, one to handle logins from nonprivileged users only and one to handle logins from privileged users.

As shown in [Figure 4-1](#), one way of handling privileged and nonprivileged services is to define the default TELNET login service as privileged and a second service as nonprivileged. If security is an issue (for example, when you want to control access by assigning only a select number of ports that are privileged and assigning all others nonprivileged), you should consider changing the default TELNET login service to be nonprivileged (especially if you have connections to the Internet), and creating a privileged service that takes advantage of the `-in_ip` parameter in the device entry to specify an IP address and a range of ports.

[Figure 4-3](#) shows the entries needed in the `devices.tin` file, the `telnet service` database file, and the `services` database file for a default login service that is nonprivileged and a second service that is privileged. Important information is shown in boldface.

The following list summarizes the information in the entries.

- The first device entry, which is clonable, accommodates a nonprivileged login service that represents the default TELNET login service.
- The second device entry, which is nonclonable, accommodates a privileged login service that uses the `-in_ip` parameter. This login device will be used if the remote host has the address 172.16.3.12 with the port number between 2100 and 2150.
- The entire device name specified in the device entry for each service matches the device prefix specified for each service in the `telnet service` database file (the last field).

- The values specified in the `login_slave` and `priv_terminal` fields of the device entry for each service match the values specified for the `login_slave` and `privileged` arguments in the `telnet-service` database file.
- The service name specified for each login service in the `telnet-service` database file (the first field) matches the service name specified for each service in the `services` database file.

**Figure 4-3. Nonprivileged Service and a Service with `-in_ip`****devices.tin File Information**

```
/* clonable device entry for a default nonpriv service */
/  =name          tli_logm1    /* matches last field of telnet-service */
  =module_name    m1
  =terminal_type  ascii
  =device_type    window_term
  =login_slave    1
  =priv_terminal  0 /* changed to create a default nonpriv service */
  =dialup         0
  =clone_limit    1000

/* Nonclonable device entry for a priv login service using -in_ip */
  =name tli_inipm1.1
  =module_name    m1
  =terminal_type  ascii
  =device_type    window_term
  =login_slave    1
  =priv_terminal  1
  =dialup         0
  =parameters     '-in_ip 172.16.3.12,2100-2150'
```

**telnet-service File Information**

```
telnet window_term "keepalive nodelay" "Default Telnet login" 1 0 tli_logm1
mliplog window_term "keepalive nodelay" "inip login" 1 1 tli_inipm1
```

**services File Information**

```
telnet      23/tcp      telnetd    # alias
mliplog     940/tcp
```

To create a device entry for a login device, specify information in the fields as follows:

► **name**

Specify the name of the clonable login device associated with a TELNET login service, or a name that uses a device prefix to define a set of login devices associated with a login service. For ease of reference, you may want to use `tli` as the first three characters of the device names for the STCP TELNET devices, and you should identify the module at the end of the clonable device name or device prefix. Avoid ambiguous naming situations.

- ▶ `module_name`  
Specify the name of the module that provides the login device.
- ▶ `terminal_type`  
Specify the type of terminal supported by the login device. Stratus recommends that you specify the value `ascii`, which enables a `telnetd` daemon process to recognize any type of remote ASCII terminal.

For information about changing the terminal type of a login device after having logged in to OpenVOS, see the description of the `set_terminal_parameters` command in the *OpenVOS Commands Reference Manual* (R098).

- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `window_term`, since the STCP TELNET login and slave devices rely on the window terminal software and are classified as types of window terminal devices.
- ▶ `login_slave`  
Specify the value `1` to configure a login device. (The value `0` configures a slave device.)

#### NOTE \_\_\_\_\_

When you specify the value `1` for the `login_slave` field, you must also make sure that the value `1` applies to the entry for the login service in the `telnetSERVICE` database file.

- ▶ `priv_terminal`  
Specify the value `1` to have the login device associated with a TELNET login service support **both** nonprivileged and privileged users; specify the value `0` to have the login device support nonprivileged users only. All login devices associated with a specific TELNET login service must use the same value in the `priv_terminal` field. (The TELNET login service defined in the template `telnetSERVICE` file specifies the value `1`.) If you specify the value `1` in the `priv_terminal` field, you must also specify the value `1` in the entry for the TELNET login service in the `telnetSERVICE` database file, as described in [“How to Edit the telnetSERVICE File” on page 5-55](#). If you specify the value `0`, you must also specify the value `0` in the entry for the TELNET login service in the `telnetSERVICE` database file.
- ▶ `parameters`  
Specify arguments that establish the parameters associated with the login device. Login devices (specifically, incoming, non-clonable login devices) have one optional argument, `-in_ip`.



You can optionally use the `-in_ip` argument to specify the 4-byte IPv4 address of a remote terminal device that will use STCP TELNET to request a login connection, which thereby enables you to control which STCP TELNET login device on the module will service the incoming request. When the STCP TELNET server receives the incoming request, it selects the TELNET login device whose `-in_ip` value in the `parameters` field matches the IPv4 address of the remote terminal device making the request, if a match exists. You must use the standard dot notation when specifying the IPv4 address (for example, `172.16.2.12`).

In addition to specifying the IPv4 address, you can use the `-in_ip` argument to specify a network port number or a range of port numbers, with an optional bit mask. Using a network port number or range of port numbers is useful for handling incoming login requests from remote terminal devices that are controlled by a terminal server. In this situation, the IPv4 address specifies the terminal server, and the network port number (or range of port numbers) specifies a valid port for the actual remote terminal device making the request. This creates a one-to-one relationship between a TELNET login device and a remote terminal, even though the remote terminal is controlled by a terminal server. Note that if you specify a range, use the format `l-h`, where `l` is the lowest number in the range and `h` is the highest. A range is useful if you cannot predict the terminal that will request the connection.

If you specify a bit mask, it is applied to the incoming port number before it is compared to the first and last port numbers in a range of port numbers. The bit mask can be in decimal or hexadecimal format and is terminated by an optional `x` or `X` character.

The `-in_ip` argument therefore has four possible formats. (In the last format, note that `port_mask` refers to the bit mask.)

```
-in_ip ip_address
-in_ip ip_address,port_number
-in_ip ip_address,port_number_range
-in_ip ip_address,port_number_range;port_mask
```

You can use the `-in_ip` argument only with nonclonable configurations.

## NOTES \_\_\_\_\_

1. You must enclose in apostrophes the values that you specify in the `parameters` field.
2. The TLI access-layer device driver is the default access-layer device driver, so you do not need to specify the value `tli_al` for the `-access_layer` argument.

► `dialup`

Specify the value 1 if you want the login connection to be treated as a dialup connection, which imposes an inactivity timeout. The default is 0, which specifies that the device is a nondialup device. Note that OpenVOS terminates the login process if it loses the TCP connection, regardless of this setting.

For information about specifying the number of minutes that a process running on a dialup or nondialup line can be inactive before it is terminated, see the description of the `logout_admin` command in the manual *OpenVOS System Administration: Registration and Security* (R283).

► `clone_limit`

Specify the clonable limit for the type of login device, such as 1000, which handles up to 1000 connections (you may want to change this value based on your network requirements). With clonable device entries, you need to create only one login device entry for a particular login service. The value 1000 can be applied to login devices; however, the maximum value for the clone limit is 8120.

## Defining STCP TELNET Incoming Slave Devices

An incoming slave device enables a remote user to use TELNET to connect to an available service (active OpenVOS application) on the local module. To enable the `telnetd` daemon process to service these client-initiated connection requests for incoming slave services on the local module, you must perform the following steps.

1. Determine how many incoming slave services you need to advertise on the local module and the options that should be associated with each of these services.
2. Select a method for creating your STCP TELNET incoming slave device entries and create the entries. Select one of two methods, as follows:
  - Create a single clonable device entry for each incoming slave service. In this case, the entire device name of the clonable device (for example, `tli_inslv1m1`) serves as the device prefix for the incoming slave service in the `telnet-service` file (or the `-device_prefix` specified with the `telnet_admin` command, if you are adding a service after the stack is up). The clone limit enables TELNET to automatically create incoming slave devices for an incoming slave service when it needs them.
  - As an alternative, create all of the device entries for a particular incoming slave service individually and use a common prefix of the device name to group the entries (for example, `tli_inslv2m1.1`, `tli_inslv2m1.2`, and `tli_inslv2m1.3`). You would then specify this device prefix in the `telnet-service` file (or with `telnet_admin`).
3. Create a service entry for each incoming slave service by either editing the `telnet-service` and `services` database files or by issuing the `telnet_admin` command, which updates both files and is used when the stack is already running

on the module. You should consider specifying the `keepalive` and `no-delay` options for your incoming slave services.

4. Make sure that the applications can attach to and open the incoming slave devices. (An application must attach to and open a port to one or more of its associated incoming slave devices and wait for a remote user to initiate a connection request for the slave service. The `telnetd` daemon process uses the clonable device name or the prefix and selects an incoming slave device that has already been attached to and opened by the application running on the module.) The method of attaching applications to an incoming slave device depends on whether you are using a clonable device entry or individual device entries with a common prefix for the service.
  - If you are using a clonable device entry, each `s$attach_port` call for a service must specify the same device name (for example, three calls would all specify `#tli_inslv1m1`).
  - If you are specifying individual device entries, each `s$attach_port` call for a service must specify a device name that includes the appropriate device prefix but has a unique suffix (for example, one call would specify a device name such as `#tli_inslv2m1.1`, another call would specify `#tli_inslv2m1.2`, and a third call would specify `#tli_inslv2m1.3`). (“[Enabling Incoming \(Client-Initiated\) Connections](#)” on page C-1 describes how to attach applications to incoming slave devices.)

[Figure 4-4](#) shows the entries needed in the `devices.tin` file, the `telnetSERVICE` database file, and the `SERVICES` database file for two incoming slave services. Important information is shown in boldface. The following list summarizes the information in the entries.

The `telnetSERVICE` file also defines the service name of the slave service (the first field), which must match the service name specified for the service in the `SERVICES` file.

- The first clonable device entry accommodates an incoming slave service that has the device name `tli_inslv1m1`.
- The second clonable device entry accommodates an incoming slave service that has the device name `tli_inslv2m1`.
- The entire device name specified in the device entry for each incoming slave service matches the device prefix specified for each service in the `telnetSERVICE` database file (the last field).
- The value specified in the `login_slave` field of the device entry for each incoming slave service (the value 0) matches the value specified for the `login_slave` argument in the `telnetSERVICE` file.
- Note that although the concept of privileged and nonprivileged does not have any relevance with slave devices, the value specified in the `priv_terminal` field of

the device entry for each service must match the value specified for the *privileged* argument in the `telnet-service` file.

- The service name specified for each incoming slave service in the `telnet-service` database file (the first field) matches the service name specified for each service in the `services` database file.

Note that if you specify a clonable device entry for a service, issuing the `list-devices` command with the `-type window-term` argument lists the cloned TLI slave devices (for example, `tli_inslv1m1_1` and `tli_inslv1m1_2`).

#### Figure 4-4. Incoming Slave Services with Clonable Device Entries

##### `devices.tin` File Information

```
/* clonable device entry for the first incoming slave service */
/  =name          tli_inslv1m1 /* matches last field of telnet-service */
  =module_name    m1
  =terminal_type  ascii
  =device_type    window-term
  =login_slave    0
  =priv_terminal  0
  =dialup         0
  =clone_limit    1000
```

```
/* clonable device entry for a second incoming slave service */
/  =name          tli_inslv2m1
  =module_name    m1
  =terminal_type  ascii
  =device_type    window-term
  =login_slave    0
  =priv_terminal  0
  =dialup         0
  =clone_limit    1000
```

##### `telnet-service` File Information for the Slave Services

```
m1slave1 window-term "keepalive nodelay" "slave1 service" 0 0 tli_inslv1m1
m1slave2 window-term "keepalive linger=10 nodelay" "slave2 linger" 0 0
+tli_inslv2m1
```

##### `services` File Information for the Slave Services

```
m1slave1      950/tcp
m1slave2      951/tcp
```

Figure 4-5 shows one incoming slave service with individual device entries instead of clonable device entries in the `devices.tin` file, an entry in the `telnet-service` database file, and an entry in the `services` database file. Important information is shown in boldface. The following list summarizes the information in the entries.

- The device entries use the same prefix (`tli_inslv1m1`) but different suffixes (`.1` and `.2`) to define slave devices for the incoming slave service.
- The device prefix specified in both device entries matches the device prefix specified for the slave service in the `telnet-service` database file (the last field).
- The value specified in the `login_slave` field of the device entry for the incoming slave service (the value `0`) matches the value specified for the `login_slave` argument in the `telnet-service` file.
- Note that although the concept of privileged and nonprivileged does not have any relevance with slave devices, the value specified in the `priv_terminal` field of the device entry must match the value specified for the `privileged` argument in the `telnet-service` database file.
- The service name specified for the slave service in the `telnet-service` database file (the first field) matches the service name specified for the slave service in the `services` database file.

**Figure 4-5. Incoming Slave Service with Individual Device Entries**

**devices.tin File Information**

```
/* First device for the m1slave1 service */
/  =name          tli_inslv1m1.1 /* prefix appears in telnet-service */
    =module_name   m1
    =terminal_type  ascii
    =device_type    window_term
    =login_slave    0
    =priv_terminal  0
    =dialup         0

/* Second device for the m1slave1 service */
/  =name          tli_inslv1m1.2
    =module_name   m1
    =terminal_type  ascii
    =device_type    window_term
    =login_slave    0
    =priv_terminal  0
    =dialup         0
```

**telnet-service File Information**

```
m1slave1 window_term "keepalive nodelay" "slave1 service" 0 0 tli_inslv1m1
```

**services File Information**

```
m1slave1 950/tcp
```

Table 4-2 describes the relationship between the fields in the device entry for an incoming slave service and the fields in the `telnet` database file (and with the command arguments that can be issued with the `telnet_admin` command).

**Table 4-2. Key Values for Defining Incoming Slave Devices**

<b>devices.tin Field</b>	<b>telnet File Argument or telnet_admin Argument</b>	<b>Explanation</b>
name	<i>device_prefix</i> or <i>-device_prefix</i>	In a clonable device entry for an incoming slave service, the entire name that you specify in the <code>name</code> field must match the name that you specify for the <i>device_prefix</i> argument in the <code>telnet</code> database file (or for the <i>-device_prefix</i> argument of the <code>telnet_admin</code> command). If you specify individual device entries for an incoming slave service, each entry for the service must have the same prefix in the <code>name</code> field, and the prefix must match the prefix that you specify for the <i>device_prefix</i> argument in the <code>telnet</code> database file (or for the <i>-device_prefix</i> argument of the <code>telnet_admin</code> command). The name or prefix should identify the module. For ease of reference, you may want the names of all STCP TELNET services to include <code>tli</code> .
login_slave	<i>login_slave</i> or <i>-no_login</i>	Specify the value 0 in the <code>login_slave</code> field and the value 0 for the <i>login_slave</i> argument in the <code>telnet</code> database file (or the <i>-no_login</i> argument if you issue the <code>telnet_admin</code> command).
priv_terminal	<i>privileged</i> or <i>-privileged</i>	Although the privileged/nonprivileged setting does not affect the behavior of slave devices, you must make sure that the <code>priv_terminal</code> field value matches the value specified for the <i>privileged</i> argument in the <code>telnet</code> database file. For example, if you specify the value 1 in the <code>priv_terminal</code> field, specify a 1 for the <i>privileged</i> argument of the <code>telnet</code> database file (or specify the <i>-privileged</i> argument of the <code>telnet_admin</code> command). If you specify a 0 in the <code>priv_terminal</code> field, specify a 0 for the <i>privileged</i> argument of the <code>telnet</code> database file (or specify the <i>-no_privileged</i> argument of the <code>telnet_admin</code> command).

To create a device entry for an incoming slave device, specify information in the fields as follows:

- ▶ `name`  
Specify the name of the clonable incoming slave device associated with a local TELNET slave service, or a name that uses a device prefix to define a set of incoming slave devices associated with a local slave service. For ease of reference, you may want to use `tli` as the first three characters of the device names for the STCP TELNET devices, and you should identify the module at the end of the clonable device name or device prefix. Avoid ambiguous naming situations. (No two device entries that you create can use exactly the same name.)

#### NOTE

In order for an application to access the device that you defined in this entry, it must attach to the device using the entire clonable device name or the prefix of the name that you specify in the `name` field. If you are using a clonable device entry, the appropriate application (service) must attach and open a port to an incoming slave device using the entire clonable device name. This means that each `s$attach_port` call must specify the same clonable device name (for example, three calls would all specify `tli_inslv1m1`). (For more information about how applications attach to slave devices, see [“Enabling Incoming \(Client-Initiated\) Connections” on page C-1.](#))

- ▶ `module_name`  
Specify the name of the module that provides the incoming slave device.
- ▶ `terminal_type`  
Specify the terminal type supported by the slave device. Specify the value `ascii` to enable the `telnetd` daemon process to recognize a remote ASCII device.
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `window_term`, since the TELNET devices rely on the window terminal software and are classified as types of window terminal devices.
- ▶ `login_slave`  
Specify the value `0` to configure a slave device. (The value `1` configures a login device.)
- ▶ `priv_terminal`  
Although the distinction of nonprivileged and privileged users does not apply to incoming slave devices, the value you specify in this field must match the value you

specify in the `telnet` service database file for the incoming slave service. If you specify the value 0 in the `priv_terminal` field, you must also specify the value 0 in the entry for the incoming slave service in the `telnet` service database file, as described in “[How the telnet](#) service File Is Used” on page 5-53. If you specify the value 1, you must also specify the value 1 in the entry for the incoming slave service in the `telnet` service database file.

► `clone_limit`

Specify the clonable limit for the type of incoming slave device, such as 1000, which handles up to 1000 connections (you may want to change this value based on your network requirements). With clonable device entries, you need to create only one incoming slave device entry for a range of slave devices associated with a given service. The value 1000 can be applied to slave devices; however, the maximum value for the clone limit is 8120.

## Defining STCP TELNET Outgoing Slave Devices

Outgoing slave devices are logical devices defined on the local module to handle host-initiated connection requests for remote services. With a host-initiated connection, an application running on the local STCP module (such as the OpenVOS spooler process) attaches and opens a port to an outgoing slave device associated with a remote service (for example, a remote network printer); the local `telnetd` daemon process then initiates the connection on behalf of the application.

To enable `telnetd` to use outgoing slave devices to connect to remote services, you must perform the following steps.

1. Determine how many different remote services you need to support with local outgoing slave devices. Each outgoing slave device defined in the local module's `devices.tin` file refers to a remote service available on the network, and a remote configuration file (such as the `telnet` service and/or `services` database file, depending on what is running remotely), **not** the local module's `telnet` service and `services` files, must define the service (that is, the service must be defined remotely). The local module's `telnet` service file should only contain entries that handle incoming connections (login and incoming slave connections), and the `telnet_admin` command applies only to login and incoming slave connections.
2. Select a method for creating your STCP TELNET outgoing slave device entries and create the entries. Select one of two methods, as follows:
  - Create a single clonable device entry that represents a remote service. In this case, the entire device name (for example, `tli_outslv1m1`) of the clonable device is used locally to represent the remote service. The clone limit enables the `telnetd` process to automatically create outgoing slave devices whenever it needs to connect to the remote service.



- As an alternative, create all of the outgoing slave device entries for a particular remote service individually and use a common prefix of the device name to group the entries (for example, `tli_outslv2m1.1`, `tli_outslv2m1.2`, and `tli_outslv2m1.3`).
3. Make sure that a local application can attach to and open the outgoing slave devices. The method of attaching an application to an outgoing slave device depends on whether you are using a clonable device entry or individual device entries with a common prefix.
- If you are using a clonable device entry, each `s$attach_port` call for a service must specify the same device name (for example, three calls would all specify `#tli_outslv1m1`).
  - If you are specifying individual device entries, each `s$attach_port` call for a service must specify a device name that includes the appropriate device prefix but has a unique suffix (for example, one call would specify a device name such as `#tli_outslv2m1.1`, another call would specify `#tli_outslv2m1.2`, and a third call would specify `#tli_outslv2m1.3`). (“[Enabling Outgoing \(Host-Initiated\) Connections](#)” on page C-5 describes how to attach applications to outgoing slave devices.)

For additional information, see the following sections:

- “[Field Descriptions for Clonable Outgoing Slave Device Entries](#)” on page 4-27
- “[Sample Clonable Outgoing Slave Device Entry for a Printer](#)” on page 4-29
- “[Sample Clonable Outgoing Slave Device Entry for a Remote Application](#)” on page 4-29

### Field Descriptions for Clonable Outgoing Slave Device Entries

To create a device entry for an outgoing slave device, specify information in the fields as follows:

- **name**
- Specify a name for the outgoing slave device. (No two devices can have exactly the same name.)

#### NOTE

For an application to access the device that you define in this entry, it must attach to the device using the clonable device name or the prefix of the name that you specify in the `name` field. (For more information about how applications attach to slave devices, see “[Enabling Outgoing \(Host-Initiated\) Connections](#)” on page C-5.)

- ▶ `module_name`  
Specify the name of the module that provides the outgoing slave device.
- ▶ `terminal_type`  
Specify the terminal type supported by the slave device. Specify the value `ascii` to enable the `telnetd` daemon process to recognize a remote ASCII device.
- ▶ `device_type`  
Specify the type of device that you are configuring. You must specify the value `window_term`, since the TELNET devices rely on the window terminal software and are classified as types of window terminal devices.
- ▶ `login_slave`  
Specify the value 0 to configure a slave device. (The value 1 configures a login device.)
- ▶ `parameters`  
Specify the arguments that establish the parameters associated with the outgoing slave device. Enclose in apostrophes ( ' ' ) the values that you specify for this field.
  - You must specify the `-ip` argument and identify both the IPv4 address and the network port number of the remote client to which the device will connect. You must use the standard dot notation when specifying the IPv4 address. The `telnetd` daemon process sends a TCP/IP connection request to the port specified in this IP address. Separate the IPv4 address and the port number with a comma ( , ) without additional spaces.
  - For printers, specify `-tcp_only`.
  - For devices that are not printers, specify `-bits_per_char 8`.
  - For outgoing slave devices that specify an IPv4 address of a remote server that is TCP-based and not TELNET-based, you must also specify the argument `-tcp_only`. This argument suppresses the TELNET protocol for this connection and causes the connection to operate in 8-bit mode.
- ▶ `clone_limit` or `clone_limit_32`  
Specify the clonable limit for the type of outgoing slave device, such as 1000, which handles up to 1000 connections (you may want to change this value based on your network requirements). With clonable device entries, you need to create only one outgoing slave device entry for a range of slave devices associated with a given service. For module running OpenVOS Release 17.2.0 (or later), use the `clone_limit_32` field; for modules running earlier releases, use the `clone_limit` field. The value 1000 can be applied to slave devices; however, check the field descriptions for the maximum values.

For example entries, see the following sections:

- [“Sample Clonable Outgoing Slave Device Entry for a Printer” on page 4-29](#)
- [“Sample Clonable Outgoing Slave Device Entry for a Remote Application” on page 4-29](#)

### Sample Clonable Outgoing Slave Device Entry for a Printer

The following sample entry defines the clonable outgoing slave device #tli\_outslv1m1 as a printer. In the example, STCP TELNET uses the outgoing slave device to connect to a printer (a service) advertised by module #m1, represented remotely by the service name m1printer1 and the IP address 172.16.3.25 at port 958. In this case, #m1 has a telnetservice file (and/or services file) that advertises this service at a given port number.

```
/    =name          tli_outslv1m1
    =module_name    m1
    =terminal_type  ascii
    =device_type    window_term
    =login_slave    0
    =parameters     '-tcp_only -ip 172.16.3.25,958'
    =clone_limit     1000
```

### Sample Clonable Outgoing Slave Device Entry for a Remote Application

The following sample entry defines the clonable outgoing slave device #tli\_outslv2m1 for a remote application. In the example, STCP TELNET uses the outgoing slave device to connect to a remote application (a service) advertised by module #m2, represented remotely by the service name m2app1 and the IP address 172.16.3.26 at port 959. In this case, #m2 has a telnetservice file (and/or services file) that advertises this service at a given port number. (To support the remote service, local module #m1 simply needs the device entry in the devices.tin file.)

```
/    =name          tli_outslv2m1
    =module_name    m1
    =terminal_type  ascii
    =device_type    window_term
    =login_slave    0
    =parameters     '-ip 172.16.3.26,959 -bits_per_char 8'
    =clone_limit     1000
```

The corresponding entries for the incoming slave device in the `devices.tin`, `telnetSERVICE`, and `SERVICES` files on the remote system are, as follows (do not enter the plus sign (+) in the `telnetSERVICE` file entry because it is a line-continuation character):

#### **devices.tin File Information**

```
/      =name          tli_inslv2m2
      =module_name    m2
      =terminal_type  ascii
      =device_type    window_term
      =login_slave    0
      =parameters     '-bits_per_char 8'
      =clone_limit     1000
```

#### **telnetSERVICE File Information**

```
m2app1 window_term "keepalive nodelay" "Telnet Incoming Slave" 0
+ 0 tli_inslv2m2
```

#### **SERVICES File Information**

```
m2app1 959/tcp telnetd # alias
```

## **Creating, Installing, and Activating the `devices.table` File**

After you define the required STCP devices in the `devices.tin` file, you must issue the `create_table` command from the directory `(master_disk)>system>configuration` to create a new `devices.table` file, as follows:

```
create_table devices
```

For additional information about creating, installing, and activating the `devices.table` file, see [“Overview of Device Configuration” on page 3-1](#).

## **Controlling Access to STCP Devices**

You can control access to the STCP drivers and other STCP devices whose `device_type` value is `streams` or `streams_pci`. To do so, create a device access control list (ACL) file in the `(master_disk)>system>acl` directory and give this file the permissions that you want for the STCP devices. You then specify the name of the device ACL file in the `access_list_name` field of the `devices.tin` file entries for the STCP devices whose access you are controlling. The examples in this section use the file name `stcp_access`, but you can use any name.

In addition, many TCP/IP server applications (and some client applications) assume, by convention, that applications using port numbers less than 1024 are privileged processes, and applications using port numbers greater than 1024 are non-privileged processes. By using the ACL for the device ACL file, you can prevent non-privileged users from binding to ports below 1024.

#### NOTE

In this section, *privileged* refers to users who are allowed to bind to ports below 1024 because they have write access to the device ACL file, and *non-privileged* refers to users who cannot bind to ports below 1024 because they do not have write access to the device ACL file. So, in this context, *privileged* is unrelated to an OpenVOS privileged user.

The STCP administrator (typically, `*.SysAdmin`) should have write access to the device ACL file (located in the `(master_disk)>system>acl` directory). All other STCP users (typically, `*.*`) should have read access to this file.

To implement this access, perform the following steps, which give all users (`*.*`) read access and give privileged users (for example, `*.SysAdmin`, `Privileged_user.*`, and `*.other_privileged_group`) write access to STCP:

1. Create the device ACL file (in the example, the device ACL file is `stcp_access`) in the `(master_disk)>system>acl` directory by issuing the following commands:

```
change_current_directory (master_disk)>system>acl
create_file stcp_access
```

2. Issue the `give_access` command, specifying the name of the device ACL file for the `path_name` argument.

For example, issue the following command to give non-privileged access to devices whose `devices.tin` file entries include `access_list_name stcp_access`:

```
give_access read (master_disk)>system>acl>stcp_access -user *.*
```

For example, issue the following command to give privileged access to devices whose `devices.tin` file entries include, for example, `access_list_name stcp_access` (do not enter the plus sign (+) because it is a line-continuation character):

```
give_access write (master_disk)>system>acl>stcp_access -user
+ *.SysAdmin
give_access write (master_disk)>system>acl>stcp_access -user
+ *.other_privileged_group
give_access write (master_disk)>system>acl>stcp_access -user
+ Privileged_user.*
```

3. Include the `access_list_name` field in the `devices.tin` file entries for STCP devices whose `device_type` value is `streams` or `streams_pci`. For examples, see [“Defining STCP Protocol Drivers” on page 4-2](#), [“Defining STCP TELNET Devices” on page 4-8](#), or [“Configuring SDLMUX Devices” on page 3-12](#).
4. Activate the newly modified `devices.tin` file by issuing the following commands:

```
create_table devices.tin
broadcast_file devices.table >system
configure_devices
```

For additional information about access, see the manual *OpenVOS System Administration: Registration and Security* (R283).

---

## Chapter 5

# Configuring the STCP Database Files

As part of the configuration procedures for STCP, you must provide information about your network configuration in a number of database files. These database files describe the various aspects of your network configuration and enable STCP to reference host and network names and addresses, identify services and protocols used by your system, perform network routing, and configure network interfaces within the STCP stack. By supplying information in the files, you enable users to access hosts on the local network and other supported networks and use the various protocols and services available.

This chapter, which contains the following sections, describes how the information in the STCP database files is used and how to edit the files.

- [“Overview of the STCP Template Database Files” on page 5-2](#)
- [“The `bootptab` File and Boot-Related Information” on page 5-4](#)
- [“The `hosts` File and Host Name Information” on page 5-7](#)
- [“The `hosts.allow` and `hosts.deny` Files and TCP Wrappers” on page 5-10](#)
- [“The `inetd.conf` File and Information for `inetd`” on page 5-21](#)
- [“The `networks` File and Network Information” on page 5-26](#)
- [“The `nsswitch.conf` File and Name Server Information” on page 5-28](#)
- [“The `ospfdconf` File and Routing Information” on page 5-31](#)
- [“The `protocols` File and Protocol Information” on page 5-40](#)
- [“The `resolv.conf` File and Name Server Information” on page 5-42](#)
- [“The `services` File and Service Information” on page 5-46](#)
- [“The `snmpconf` File and SNMP Information” on page 5-49](#)
- [“The `telnet-service` File and TELNET Information” on page 5-53](#)
- [“The `xinetd.conf` File and Information for `xinetd`” on page 5-57](#)

# Overview of the STCP Template Database Files

The STCP template database files reside in the `(master_disk)>system>stcp>templates` directory. These template files show the basic format required to build your configuration-specific STCP database files.

Do **not** edit the original STCP template files; instead, use the `copy_file` command to make a copy of each template file. Edit the copies to suit your configuration and place the final versions in the `(master_disk)>system>stcp` directory so that they are available to STCP. (For a description of the `copy_file` command, see the *OpenVOS Commands Reference Manual* (R098).)

Table 5-1 describes the STCP database files in alphabetical order.

Table 5-1. STCP Database Files (Page 1 of 3)

Database File	Description
<code>bootptab</code>	Supplies information that enables the <code>bootpd</code> daemon process to service boot-file requests. If the module will service boot-file requests, supply boot information in this file. For complete information, see “ <a href="#">The bootptab File and Boot-Related Information</a> ” on page 5-4.
<code>hosts</code>	Defines the host names and Internet addresses (IPv4 addresses) of hosts, both local and remote, to which you will connect using STCP. For complete information, see “ <a href="#">The hosts File and Host Name Information</a> ” on page 5-7.
<code>hosts.allow</code> and <code>hosts.deny</code>	Supplies information that enables certain services (FTP, TELNET, and services that the <code>inetd</code> daemon starts) to authenticate and log client requests as part of TCP wrappers functionality. In the <code>hosts.allow</code> file, you list the path names of the daemons for services to which you want to provide client access, and you list the names of the clients to which you want to give access. In the <code>hosts.deny</code> file, you list the names of clients and the path names of the daemons for services to which you want to deny client access. For complete information, see “ <a href="#">The hosts.allow and hosts.deny Files and TCP Wrappers</a> ” on page 5-10.
<code>inetd.conf</code>	Supplies information that enables the <code>inetd</code> daemon process to service incoming connection requests and start the respective daemon processes (for example, <code>bootpd</code> and <code>tftpd</code> , if they are explicitly activated). Make sure that this file defines all STCP <code>inetd</code> services that are available on the module. For complete information, see “ <a href="#">The inetd.conf File and Information for inetd</a> ” on page 5-21.
<code>networks</code>	Supplies the addresses, names, and aliases of all networks to which you connect using STCP. Although it is not mandatory that you customize this file, it does help you track the networks and/or subnets in your configuration. For complete information, see “ <a href="#">The networks File and Network Information</a> ” on page 5-26.



**Table 5-1. STCP Database Files** (Page 2 of 3)

Database File	Description
<code>nsswitch.conf</code>	Specifies the name services and their search order, which a system uses for lookups that use the name server switch (nsswitch) facility. Though not a database file, you need to configure this file when you configure relevant database files. This file is used only by programs that have been built under POSIX rules in OpenVOS Release 17.1 or later. For complete information, see <a href="#">“The <code>nsswitch.conf</code> File and Name Server Information” on page 5-28</a> .
<code>omonconf</code>	Lists the routers that you want to monitor from this module. For additional information, see the description of the <code>omon</code> command in <a href="#">Chapter 9</a> .
<code>ospfdconf</code>	Supplies the routing information that enables an OpenVOS module to participate in an Open Shortest Path First (OSPF) autonomous system (AS). To run the <code>ospfd</code> daemon process, edit this file to provide information that is specific to your network. For complete information, see <a href="#">“The <code>ospfdconf</code> File and Routing Information” on page 5-31</a> .
<code>protocols</code>	Supplies information about the supported protocols. You typically do not have to customize this file, but a version of this file must be available to STCP. For complete information, see <a href="#">“The <code>protocols</code> File and Protocol Information” on page 5-40</a> .
<code>resolv.conf</code>	Identifies the name servers available to resolve host names and IP addresses. If your configuration relies on name servers, customize this file. For complete information, see <a href="#">“The <code>resolv.conf</code> File and Name Server Information” on page 5-42</a> .
<code>services</code>	Supplies information about the services (for example, TELNET and FTP) that are supported by STCP on the module. Your version of this file must define each service that is supported by STCP, including any services that you have added, such as any incoming TELNET slave services. For complete information, see <a href="#">“The <code>services</code> File and Service Information” on page 5-46</a> .
<code>snmpconf</code>	Supplies the <code>snmpd</code> daemon process with site-specific administrative information. Although it is not mandatory that you customize this file, you may want to use it to provide useful administrative information, such as a contact person. For complete information, see <a href="#">“The <code>snmpconf</code> File and SNMP Information” on page 5-49</a> .
<code>sync_cfgd.conf</code>	Supplies information to the <code>sync_cfgd</code> daemon, which is a Dynamic Host Configuration Protocol (DHCP) server for network I/O enclosure devices. The <code>sync_cfgd</code> daemon automatically assigns IP addresses to these devices based on the information in the <code>sync_cfgd.conf</code> file. For complete information, see the <i>Stratus fitServer: Network I/O Enclosure Guide</i> (R608).

Table 5-1. STCP Database Files (Page 3 of 3)

Database File	Description
telnet.service	Specifies information about the local TELNET services supported by the <code>telnetd</code> daemon process. Customize this file to change characteristics of the default TELNET login service or define additional local TELNET services, such as local slave services. For complete information, see “ <a href="#">The telnet.service File and TELNET Information</a> ” on page 5-53.
xinetd.conf	Configures the <code>xinetd</code> daemon process. For information, see “ <a href="#">The xinetd.conf File and Information for xinetd</a> ” on page 5-57.

The remainder of this chapter provides detailed descriptions of each database file (except the `sync_cfgd.conf` file). Note that the figures showing the template files do not include the individual copyright information. See the actual template files to examine copyright information.

## The bootptab File and Boot-Related Information

The `bootptab` file provides boot-related information that enables STCP on the module to respond to boot requests from other hosts on the network. It contains the name of a default boot directory and file, as well as the host names and addresses of the hosts that can make STCP boot requests.

The following sections provide additional information about the `bootptab` file.

- “[How the bootptab File Is Used](#)” on page 5-4
- “[How to Edit the bootptab File](#)” on page 5-5
- “[Sample bootptab File](#)” on page 5-6
- “[Related Information](#)” on page 5-7

### How the bootptab File Is Used

The `bootptab` file supports the `bootpd` daemon process, an `inetd` service that implements the Bootstrap Protocol (BOOTP) and uses the UDP protocol to listen for and respond to boot requests. The BOOTP protocol allows a diskless host to query a server in order to ask for its own IP address, the address of a host off which it can boot, and the name of a boot file. (The `bootpd` daemon process supports only the IPv4 protocol and can be invoked by `inetd` only if you uncomment its command line in the `inetd.conf` database file.)

Edit the `bootptab` file if the local subnet (LAN segment) includes hosts that need to issue a boot request to `bootpd` on the module. For example, you might edit this file if the local subnet includes a router/switch, such as the network I/O enclosure.

Hosts that are identified in the `bootptab` file can issue a boot request that causes the STCP `inetd` daemon process to invoke `bootpd` and use the Trivial File Transfer Protocol daemon (`tftpd`) to download the corresponding boot file.

If you use `bootpd` and the `bootptab` file, provide the following information:

- an activated entry for `tftpd` in the `inetd.conf` file to start the `tftpd` daemon process (by default, this entry is commented out)
- an activated entry for `bootpd` in the `inetd.conf` file to start `bootpd` (by default, this entry is commented out)
- an activated entry for the service `tftp` in the `services` file
- an activated entry for the service `bootps` in the `services` file
- an activated entry for the service `bootpc` in the `services` file
- a host-specific configuration file for each host making a boot request in the `(master_disk)>system>bootfiles` directory

The default boot directory, `(master_disk)>system>bootfiles`, must contain configuration files for the devices issuing boot requests. Typically, each device making a boot request requires its own configuration file that contains device-specific information.

## How to Edit the `bootptab` File

The `bootptab` file is divided into two sections delimited by two percent signs (`%%`). The comment lines in the file begin with the number sign (`#`).

The first section of the `bootptab` file contains two lines that provide the boot directory and boot file information, as follows:

- The first line identifies the name of the default directory containing the boot file.
- The second line identifies the name of the default boot file. This file is used if a boot request does not specify another boot file.

To have the module handle boot requests, edit the second section of the `bootptab` file to contain an entry for each host that can make boot requests.

Each entry in the second section must be on a separate line. [Table 5-2](#) describes the fields used in each entry.

Table 5-2. Fields in the bootptab File

Field	Description
host	Identifies the name of the host that can make boot requests to <code>bootpd</code> (that is, be a client of the <code>bootpd</code> daemon process). Required.
htype	Identifies the type of hardware on the host. Set the value for this field to 1.
haddr	Identifies the 6-byte MAC address of the named host. Required.
iaddr	Identifies the IPv4 address of the named host. (The <code>bootpd</code> daemon process does not support the IPv6 protocol.)
bootfile	Identifies the name of the boot file to be used when the named host makes a boot request. If you do not specify this field, <code>bootpd</code> uses the default boot file identified in the first section of the file, unless the boot request itself specifies a different file. Optional.

Sample bootptab File

Figure 5-1 shows a sample bootptab file.

```
# bootptab
# Example bootp table file, database for bootp daemon.
#
# Blank lines and lines beginning with '#' are ignored.
#
# home directory
#
>system>bootfiles
# default bootfile
default

%%

# The remainder of this file contains one line per client
# interface with the information shown by the table headings
# below. First search is on internet address if specified in
# incoming request. If not specified, hardware address is used.
# The 'host' name is tried as a suffix for the 'bootfile'
# when searching the home directory (e.g., bootfile.host).
```

(Continued on next page)

# host	htype	haddr	iaddr	bootfile
ncd	1	00-00-A7-11-34-64	89.0.10.66	bootfile.ncd
m11	1	00-00-A8-8A-86-CB	89.0.10.64	bootfile.m11
router1	1	08-00-39-00-60-0E	89.0.5.9	router.1_0
router2	1	08-00-39-00-60-0F	89.0.5.10	router.1_0

**Figure 5-1. Template `bootptab` File**

## Related Information

For more information about `bootpd`, see the `bootpd` description in [Chapter 8](#). For more information about the `tftpd` command and the `tftpd` daemon process, see the `tftpd` description in [Chapter 6](#) and the `tftp` command description in [Chapter 9](#).

## The `hosts` File and Host Name Information

The `hosts` file provides host entries that map the fully qualified host names (and their aliases) to the respective IPv4 and/or IPv6 addresses. The following sections provide additional information about the `hosts` file.

- [“How the `hosts` File Is Used” on page 5-7](#)
- [“How to Edit the `hosts` File” on page 5-8](#)
- [“Sample `hosts` File” on page 5-9](#)
- [“Related Information” on page 5-10](#)

## How the `hosts` File Is Used

By convention, each host in a TCP/IP-based network has a unique Internet address (called an IP address) and a corresponding host name. The IP address (IPv4 or IPv6) and host name, along with any defined aliases, enable other devices to uniquely identify the host on the network. There are two ways to define IP addresses and host names: one involves the `hosts` database file and the other involves Domain Name Service (DNS) name servers and the `resolv.conf` database file. The `resolv.conf` file simply identifies the name servers, which resolve host names and addresses using DNS.

STCP supports both methods, but, by convention, queries the name servers listed in the `(master_disk)>system>stcp>resolv.conf` file first. It uses the `(master_disk)>system>stcp>hosts` file if the name server fails to resolve the host names. However, even if you decide to use name servers, you may want to keep your own `hosts` file up-to-date so that you can use it if a name server is not operational.

To use the `hosts` file, edit a copy of the template `hosts` file, located in the directory `(master_disk)>system>stcp>templates`. Save the copy in the `(master_disk)>system>stcp` directory.

Your `hosts` file can contain various types of entries, including the IPv4 and/or IPv6 addresses and names of all hosts that you want to reach on your local network, any remote hosts to which you will connect frequently, and the host (module) that you are configuring. If a host does not have an entry in this file and no name server exists to resolve host names, the host can only be accessed using the IP address. You do not have to create entries for routers/gateways in the `hosts` file unless you need to connect to them directly (for example, to perform administrative operations that manage the routers/gateways).

You can create the following types of entries in the `hosts` file.

- an entry for the required local host (loopback) interface, which allows a machine running STCP to talk to itself (by convention, it uses IP address `127.0.0.1`)
- an entry for each host that you want to communicate with on your local network (include an entry for each logical network interface available on your module)
- an entry for each remote host to which you will connect frequently

## How to Edit the `hosts` File

Create an entry for each host on a single line using the format shown in the template file `(master_disk)>system>stcp>templates>hosts`.

Separate the fields of a `hosts` file entry using any number of blank spaces or horizontal tabulation (tab) characters. To separate lines, use a carriage return. To create a comment line, begin the line with the number sign (`#`); routines that search the file ignore all characters from `#` to the end of the line.

[Table 5-3](#) describes the fields used in each entry.

**Table 5-3. Fields in the `hosts` File**

Field	Description
IP address	Specifies the Internet address (IP address in IPv4 or IPv6 format), in standard dot-notation, associated with the host. Required.
name	Identifies the name associated with a host or a router/gateway. Required.  A host name is a text string of up to 64 characters that can contain the letters of the alphabet (A through Z and a through z), the numbers 0 through 9, the hyphen (-), and the period (.). The first character of the text string must be alphanumeric. Host names are not case sensitive. (This definition of a host name is derived from RFC 952 and RFC 1123.)
alias	Identifies an alias (another name that can be used to identify the host). You can specify more than one alias, but to avoid confusion, do not create many aliases. Optional.
comment	Identifies a comment for the host entry. Optional.

When you add a host to your network configuration, update the `hosts` file to contain an entry. If you remove a host, delete its entry from the `hosts` file.

## Sample `hosts` File

Figure 5-2 shows the Internet-style template `hosts` file, which uses the standard template network numbers; do not assign these numbers to your network. You can, however, use the loopback IP address.

```
#           Sample hosts file
#
# IP address  name                alias                # comment(s)
#
127.0.0.1    localhost  loopback-host loopback lb # required
#
#
# The following entries are for demonstration purposes only.
# Please refer to the documentation for instructions on acquiring
# network numbers.
#
89.0.1.1     host1          h1
89.0.1.2     host2          h2
89.0.1.3     host3          h3
89.0.2.1     option        op
```

*(Continued on next page)*

89.0.3.71	regular	reg
89.0.4.1	metic1	m1
89.0.4.2	metic2	m2
89.0.4.2	metic3	m3
89.10.5.1	common1	co1
89.10.5.2	common2	co2
89.76.5.1	symbol	sys
89.1.1.1	xpress	x

**Figure 5-2. Template `hosts` File**

## Related Information

For information about the format of IP addresses, see [“Naming Conventions” on page A-3](#).

## The `hosts.allow` and `hosts.deny` Files and TCP Wrappers

The `hosts.allow` and `hosts.deny` files provide access information to daemons (`telnetd`, `ftpd`, and for `inetd`) that authenticate client access as part of TCP wrappers functionality. In the `hosts.allow` file, you list the path names of the daemons for services to which you want to provide client access, and you list the names of the clients to which you want to give access. In the `hosts.deny` file, you list the names of clients and the path names of the daemons for services to which you want to deny client access.

### NOTES

---

1. If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in connecting to services that use this functionality.
2. OpenVOS does not support certain features of TCP wrappers. The OpenVOS implementation of TCP wrappers does not support language extensions in the `hosts.allow` and `hosts.deny` files. Since OpenVOS does not provide or support a shell command-line interface, the OpenVOS implementation of TCP wrappers also does not support a shell command-line interface. The OpenVOS implementation of TCP wrappers also does not support the `tcpdchk` program (this program identifies common problems with the `hosts.allow`, `hosts.deny`, and `inetd.conf` files).



The following sections provide additional information about the `hosts.allow` and `hosts.deny` files.

- [“How the `hosts.allow` and `hosts.deny` Files Are Used” on page 5-11](#)
- [“How to Edit the `hosts.allow` and `hosts.deny` Files” on page 5-12](#)
- [“Sample `hosts.allow` and `hosts.deny` Files” on page 5-16](#)
- [“Related Information” on page 5-20](#)

## How the `hosts.allow` and `hosts.deny` Files Are Used

TCP wrappers functionality enables certain services (FTP, TELNET, and services that the `inetd` daemon starts) to authenticate and log client requests. This authentication and logging functionality is built into the `ftpd` daemon (for FTP requests) and the `telnetd` daemon (for TELNET requests). Services that the `inetd` daemon starts (for example, BOOTP and TFTP) require the `tcpd` daemon.

The authenticating daemons `telnetd`, `ftpd`, and `tcpd` use the `hosts.allow` and `hosts.deny` files to authenticate client access. The `hosts.allow` file lists clients with service daemons that can provide access to those clients; the `hosts.deny` file lists clients with service daemons that can deny access to those clients. The authenticating daemon first checks the `hosts.allow` file to determine if the client is listed with the appropriate service daemon. For example, when a client sends a TELNET connection request, the `telnetd` daemon checks the `hosts.allow` file for an entry listing the client with the `telnetd` daemon. As another example, when a client sends a TFTP connection request, the `tcpd` daemon checks the `hosts.allow` file for an entry listing the client with the `tftpd` daemon.

If the authenticating daemon finds the client listed in the `hosts.allow` file with the appropriate service daemon, the authenticating daemon accepts the connection request and logs a message in the `tcpdallow` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the client connected to the server.

If the authenticating daemon does not find the client listed in the `hosts.allow` file with the appropriate service daemon, the authenticating daemon checks if the client is listed in the `hosts.deny` file with the appropriate service daemon. If the authenticating daemon finds the client listed in the `hosts.deny` file with the appropriate service daemon, the authenticating daemon denies the connection request—if the client sent a TCP request, the authenticating daemon closes the connection; or if the client sent a UDP request, the packet is dropped. The authenticating daemon also sends a message to the `tcpddeny` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the connection was refused. If the client is not listed in either the `hosts.allow` file or the `hosts.deny` file, the authenticating daemon accepts the connection and sends a message to the

`tcpdallow` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the client connected to the server. By default, the daemon accepts connections.

(If the log files `tcpdallow` and `tcpddeny` become larger than 254 blocks, OpenVOS renames the files to `tcpdallow.yy-mm-dd.hh-mm` and `tcpddeny.yy-mm-dd.hh-mm`, and starts new log files. If you want to change the size at which log files become too large, you must contact the CAC.)

TCP wrappers functionality also allows additional access checking with paranoid checks. In a *paranoid check*, the authenticating daemon conducts a reverse lookup of the client's IP address (in IPv4 or IPv6 format) and host name to check that they correspond. When the IP address and host name do not correspond, the daemon refuses the connection request; when they do correspond, the daemon accepts the connection request.

## How to Edit the `hosts.allow` and `hosts.deny` Files

The `hosts.allow` and `hosts.deny` files reside in the directory `(master_disk)>system>stcp`. Template `hosts.allow` and `hosts.deny` files reside in the directory `(master_disk)>system>stcp>templates`. If the `hosts.allow` and `hosts.deny` files do not exist, the daemon treats them as empty files.

An entry for each client and service daemon is a single line in the format shown in the appropriate template file, `hosts.allow` or `hosts.deny` (both in the directory `(master_disk)>system>stcp>templates`). Each entry in the `hosts.allow` and `hosts.deny` files has the following format:

```
daemon:client
```

Use a colon (:) to separate the fields. Use a carriage return to separate lines. To create a comment line, begin the line with the number sign (#); routines that search the file ignore all characters from # to the end of the line. Use a backslash (\), with no following space, as a line-continuation character.

For samples of the template files, see [“Template `hosts.allow` and `hosts.deny` Files” on page 5-16](#). For additional examples of entries, see [“Example Entries in `hosts.allow` and `hosts.deny` Files” on page 5-17](#).

The value in the *daemon* field specifies the path name of a service daemon. The value in the *client* field specifies the name(s) of one or more hosts to which you want to allow (in the `hosts.allow` file) or deny (in the `hosts.deny` file) access to the service of the specified daemon. You can use the optional operator `EXCEPT` in either the *daemon* field or the *client* field.

## NOTE

Stratus recommends that the *hosts.deny* file on your system consist of the single entry *ALL:ALL*, and that you provide access to clients by creating entries in the *hosts.allow* file.

Table 5-4 describes possible values for the *daemon* field.

**Table 5-4. Values for the *daemon* Field in the *hosts.allow* and *hosts.deny* Files**

<i>daemon</i> Value	Description
<i>daemon_pathname</i>	The path name of the service daemon to which the authenticating daemon allows or denies the client (as specified in the <i>client</i> field) access. Specifies the full path name of the program module that implements the service daemon. For example, the program module that implements the <i>telnetd</i> daemon for the TELNET service typically has the path name <i>(master_disk)&gt;system&gt;stcp&gt;command_library&gt;telnetd.pm</i> .
<i>wildcard</i>	In the <i>daemon</i> field, <i>wildcard</i> must have the value <i>ALL</i> , which is the universal wildcard. The value <i>ALL</i> enables the authenticating daemon to allow or deny the client (as specified in the <i>client</i> field) access to all service daemons. You can optionally follow <i>ALL</i> with an IP address (in IPv4 or IPv6 format) of a NIC.
<i>operator</i>	The operator <i>EXCEPT</i> lists exceptions to what the authenticating daemon can allow or deny the client access (see Example 10).

Table 5-5 describes possible values for the *client* field.

**Table 5-5. Values for the *client* Field in the *hosts.allow* and *hosts.deny* Files** (Page 1 of 3)

<i>client</i> Value	Description
<i>host_name</i> or <i>partial_host_name</i>	<p>The <i>host_name</i> is the client to which the authenticating daemon allows or denies access to the service daemon (as specified in the <i>daemon</i> field). The <i>host_name</i> is a text string of up to 64 characters that can contain letters, the numbers 0 through 9, a hyphen (-), and a period (.). The first character of the text string must be alphanumeric. Host names are not case sensitive. The <i>partial_host_name</i> begins with a period (.). For example, the partial host name <i>.stratus.com</i> represents all <i>.stratus.com</i> host names.</p> <p>Note that an entry with a host name is effective only if the client's address can be mapped to a host name using the DNS service or the <i>hosts</i> file.</p>

**Table 5-5. Values for the *client* Field in the *hosts.allow* and *hosts.deny* Files** (Page 2 of 3)

<i>client</i> Value	Description
<i>user_name@host_name</i>	The <i>user_name</i> is a user on the client <i>host_name</i> to which the authenticating daemon allows or denies access to the service daemon (as specified in the <i>daemon</i> field). Note that an entry with a user name is effective only if the client is running the IDENT daemon, which enables the server to obtain the user name. Note also that OpenVOS does not support the IDENT daemon.
<i>IP_address</i> or <i>partial_IP_address</i>	The IP address (in IPv4 or IPv6 format) of the client to which the authenticating daemon allows or denies access to the service daemon (as specified in the <i>daemon</i> field). The <i>partial_IP_address</i> begins or ends with a period (.). For example, <i>.134.111</i> is a partial IPv4 address that represents all IPv4 addresses with the sequence <i>.134.111</i> . As another example, <i>134.111.</i> is a partial IPv4 address that represents all IPv4 addresses with the sequence <i>134.111</i> .
<i>IP_address/</i> <i>subnet_mask</i>	This pair of numbers specifies a range of IP addresses (in IPv4 or IPv6 format) for hosts to which the authenticating daemon allows or denies access to the service daemon (as specified in the <i>daemon</i> field). For example, <i>134.111.211.0/255.255.255.0</i> includes every IPv4 address in the range 134.111.211.0 through 134.111.211.255. All clients with IP addresses in this range are allowed or denied access to the service daemon (as specified in the <i>daemon</i> field).

**Table 5-5. Values for the `client` Field in the `hosts.allow` and `hosts.deny` Files** (Page 3 of 3)

<i>client</i> Value	Description
<i>wildcard</i>	<p>A <i>wildcard</i> must have one of the following values.</p> <p><code>ALL</code> enables the authenticating daemon to allow or deny all clients access to the service daemon (as specified in the <i>daemon</i> field).</p> <p><code>LOCAL</code> enables the authenticating daemon to allow or deny clients whose names do not contain a dot, access to the service daemon (as specified in the <i>daemon</i> field).</p> <p><code>PARANOID</code> enables the authenticating daemon to conduct a reverse lookup of a host name in order to determine that the host name corresponds to the appropriate IP address (in IPv4 or IPv6 format). If the host name corresponds to the appropriate IP address, the authenticating daemon can allow or deny the client access to the service daemon (as specified in the <i>daemon</i> field). Stratus recommends that you use <code>PARANOID</code> only in the <code>hosts.deny</code> file.</p> <p>The values <code>KNOWN</code> and <code>UNKNOWN</code> are also possible; however, use them cautiously. A client can resolve the wildcards <code>KNOWN</code> and <code>UNKNOWN</code> only when the client is running the identification daemon <code>IDENT</code>. Since OpenVOS does not support the <code>IDENT</code> daemon, do not use <code>KNOWN</code> and <code>UNKNOWN</code> for OpenVOS clients. In addition, a host name may be unavailable because of temporary name server problems; a network address may be unavailable because of software problems.</p> <p><code>UNKNOWN</code> enables the authenticating daemon to allow or deny any unknown user access to the service daemon (as specified in the <i>daemon</i> field).</p> <p><code>KNOWN</code> enables the authenticating daemon to allow or deny any known user access to the service daemon (as specified in the <i>daemon</i> field). A known user is one whose name <b>and</b> IP address are known. Any registered, legal user on the client system is a known user.</p>
<i>operator</i>	<p>The operator <code>EXCEPT</code> lists exceptions in a list of clients. For example, the following entry in the <code>hosts.deny</code> file enables the authenticating daemon to accept connection requests only from the client <code>134.111.211.233</code> for all services:</p> <pre>ALL:ALL EXCEPT 134.111.211.233</pre>

## Sample `hosts.allow` and `hosts.deny` Files

The following sections present the template `hosts.allow` and `hosts.deny` files as well as several example files.

- [“Template `hosts.allow` and `hosts.deny` Files” on page 5-16](#)
- [“Example Entries in `hosts.allow` and `hosts.deny` Files” on page 5-17](#)

### Template `hosts.allow` and `hosts.deny` Files

This section presents the template `hosts.allow` and `hosts.deny` files, which are located in the `(master_disk)>system>stcp>templates` directory.

[Figure 5-3](#) shows the template `hosts.allow` file.

```
#           Sample hosts.allow File
#
# This is the sample hosts.allow file. The file is a series of
# one-line statements, each with the following format:
#
#           <daemon> : <client>
#
# The value of <daemon> is the full pathname of the program that
# starts the daemon or a wildcard.
# The value of <client> is the host name (or partial host name),
# IP address (or partial IP address), or wildcard for the client
# that will be given access.
#
# Uncomment or modify the following lines to modify access.
#
# >system>stcp>command_library>telnetd.pm : ALL
# >system>stcp>command_library>ftpd.pm : ALL
```

**Figure 5-3. Template `hosts.allow` File**

[Figure 5-4](#) shows the template `hosts.deny` file.

```
#           Sample hosts.deny File
#
# This is the sample hosts.deny file. The file is a series of
# one-line statements, each with the following format:
#
#           <daemon> : <client>
#
#
```

*(Continued on next page)*

```
# The value of <daemon> is the pathname of the program that starts
# the daemon or a wildcard.
# The value of <client> is the host name (or partial host name),
# IP address (or partial IP address), or wildcard for the client
# that will be denied access.
#
# Uncomment or modify the following lines to modify access.
#
# ALL : ALL
```

**Figure 5-4. Template *hosts.deny* File**

### Example Entries in *hosts.allow* and *hosts.deny* Files

This section presents twelve examples of entries in *hosts.allow* and *hosts.deny* files.

**Example 1**—The following entries appear in the *hosts.allow* file.

```
>system>stcp>command_library>telnetd.pm : ALL
>system>stcp>command_library>ftpd.pm : ALL
```

These entries enable the authenticating daemons (in this case, the *telnetd* and *ftpd* daemons) to provide all clients access to the service daemons *telnetd* and *ftpd*.

**Example 2**—The following entries appear in the *hosts.allow* file.

```
>system>stcp>command_library>telnetd.pm : .sw.stratus.com
>system>stcp>command_library>tftpd.pm : 134.111.222.101
>system>stcp>command_library>ftpd.pm : ALL
```

These entries specify the following types of access.

- The *telnetd* daemon can provide access to all clients whose host names end in *.sw.stratus.com*. Assuming that the *telnetd* daemon was started with the value *yes* specified for the *-tcpwrapper\_check* and *-numeric* arguments, this entry forces the *telnetd* daemon to authenticate the host name rather than the IPv4 address.
- The authenticating daemon *tcpd* can provide the clients whose IPv4 address is *134.111.222.101* access to the *tftpd* daemon.
- The daemon *ftpd* can provide access to all clients.

**Example 3**—The following entry appears in the `hosts.allow` file.

```
>system>stcp>command_library>telnetd.pm : jsmith@bach.sw.stratus.com
```

This entry specifies that the `telnetd` daemon can provide access to the user `jsmith` on the client whose host name is `bach.sw.stratus.com`. Note that an entry with a host name is effective only if the client's address can be mapped to a host name using the DNS service or the `hosts` file.

**Example 4**—The following entry appears in the `hosts.allow` file.

```
>system>stcp>command_library>telnetd.pm : KNOWN@bach.sw.stratus.com
```

This entry specifies that the `telnetd` daemon can provide any known user on the client whose host name is `bach.sw.stratus.com` access to itself. Assuming that the `telnetd` daemon was started with the value `yes` specified for the `-tcpwrapper_check` argument and the value `no` specified for the `-numeric` argument, this entry forces the `telnetd` daemon to authenticate the host name rather than the IPv4 address. For additional information about forcing a daemon to authenticate a host name, see Example 2.

**Example 5**—The following entry appears in the `hosts.deny` file.

```
>system>stcp>command_library>ftpd.pm : UNKNOWN@.sw.stratus.com
```

This entry enables the `ftpd` daemon to deny access to any unknown user on any client whose host name ends in `.sw.stratus.com`. Assuming that the `ftpd` daemon was started with the value `yes` specified for the `-tcpwrapper_check` argument and the value `no` specified for the `-numeric` argument, this entry forces the `ftpd` daemon to authenticate the host name rather than the IPv4 address. For additional information about forcing a daemon to authenticate a host name, see Example 2.

Even though this example presents an entry in the `hosts.deny` file, note that Stratus recommends that you control client access using the `hosts.allow` file and that the `hosts.deny` file typically consist of the single entry `ALL:ALL`.

**Example 6**—The following entry appears in the `hosts.allow` file.

```
>system>stcp>command_library>telnetd.pm : brahms.sw.stratus.com
```

This entry specifies that the `telnetd` daemon can provide access to the client whose host name is `brahms.sw.stratus.com`. Assuming that the `telnetd` daemon was started with the value `yes` specified for the `-tcpwrapper_check` argument and the value `no` specified for the `-numeric` argument, this entry forces the `telnetd` daemon to authenticate the host name rather than the IPv4 address.



**Example 7**—The `EXCEPT` operator specifies exceptions in a list of hosts or clients. The following `hosts.allow` and `hosts.deny` files are examples.

```
#           The hosts.allow File
#
>system>stcp>command_library>tftpd.pm:134.111.221.241

#           The hosts.deny File
#
ALL:ALL EXCEPT 134.111.211.223
```

This `hosts.allow` file enables the authenticating daemon (in this case, the `tcpd` daemon because it authenticates client access for `tftp` requests) to accept requests for only the `tftpd` daemon only from the host whose IPv4 address is 134.111.221.241. The `tcpd` daemon denies requests from the host 134.111.221.241 for other services.

This `hosts.deny` file enables the `tcpd` daemon to deny requests for all service daemons from all hosts except the host whose IPv4 address is 134.111.211.223, thereby giving that host access to all daemons. This `hosts.deny` file also enables the daemons to deny requests from the host 134.111.221.241 for other services.

Although this example presents an entry in the `hosts.deny` file, Stratus recommends that the `hosts.deny` file consist of only the single entry `ALL:ALL`, and that you use the `hosts.allow` file to control client access.

**Example 8**—The following entry in the `hosts.allow` file provides an additional example of using the `EXCEPT` operator.

```
>system>stcp>command_library>tftpd.pm : ALL EXCEPT 134.111.211.223
```

This entry in the `hosts.allow` file enables the authenticating daemon (in this case, the `tcpd` daemon because it authenticates client access for `tftp` requests) to accept requests for the `tftpd` daemon from all clients except the client whose IPv4 address is 134.111.211.223. Note, though, that the `tcpd` daemon will deny the client whose IPv4 address is 134.111.211.223 access to the `tftpd` daemon only if an entry in the `hosts.deny` file explicitly specifies so.

**Example 9**—The `PARANOID` wildcard enables the authenticating daemon to conduct a reverse lookup of a host name in order to determine that the host name corresponds to the appropriate IPv4 address. If you use `PARANOID`, Stratus recommends that you use it only in the `hosts.deny` file. The entry in the following `hosts.deny` file is an example.

```
>system>stcp>command_library>telnetd.pm : PARANOID
```

This entry in the `hosts.deny` file enables the `telnetd` daemon to conduct a reverse lookup of clients. The daemon will deny access to clients that fail the paranoid check.

Although this example presents an entry in the `hosts.deny` file, Stratus recommends that the `hosts.deny` file consist of only the single entry `ALL:ALL`, and that you use the `hosts.allow` file to control client access.

**Example 10**—The following entry in the `hosts.deny` file provides an example of the `EXCEPT` operator in the `daemon` field and the `PARANOID` wildcard in the `client` field.

```
ALL EXCEPT >system>stcp>command_library>ftpd.pm : PARANOID
```

This entry in the `hosts.deny` file specifies the following.

- The `ftpd` daemon can accept requests from all clients.
- All other daemons can deny requests from all clients that fail the paranoid check but can accept requests from all clients that pass the paranoid check.

Although this example presents an entry in the `hosts.deny` file, Stratus recommends that the `hosts.deny` file consist of only the single entry `ALL:ALL`, and that you use the `hosts.allow` file to control client access.

**Example 11**—The following entry in the `hosts.allow` file provides an example of specifying a NIC in the `daemon` field.

```
ALL @192.: ALL
```

This entry in the `hosts.allow` file allows any connection to any daemon when the client connects to the IPv4 address `192.x.x.x`.

**Example 12**—The `LOCAL` wildcard enables the authenticating daemon to give any client whose name does not contain a dot access to the service daemon specified in the `daemon` field. The following `hosts.allow` and `hosts.deny` files are examples.

```
#           The hosts.allow File
#
ALL : LOCAL

#           The hosts.deny File
#
ALL:ALL
```

This `hosts.allow` file enables all daemons to accept requests from all local clients. This `hosts.deny` file enables all daemons to deny requests from all other clients.

## Related Information

For a description of the `telnetd`, `ftpd`, `tcpd`, and `inetd` daemon processes, see [Chapter 8](#). The `inetd.conf` file is described in “[The `inetd.conf` File and Information for `inetd`](#)” on page 5-21.

## The `inetd.conf` File and Information for `inetd`

The `inetd.conf` database file contains each service that you want the `inetd` daemon process to invoke when it receives an Internet connection request for that service. The following sections provide additional information about the `inetd.conf` file.

- [“How the `inetd.conf` File Is Used” on page 5-21](#)
- [“How to Edit the `inetd.conf` File” on page 5-22](#)
- [“Sample `inetd.conf` File” on page 5-24](#)
- [“Related Information” on page 5-26](#)

### How the `inetd.conf` File Is Used

The `inetd` daemon process reads the `inetd.conf` file and creates a socket for each service listed in the file. Then the `inetd` daemon process binds a port number to each socket according to the port assigned to each service in the file

(master\_disk) >system>stcp>services. Thereafter, the `inetd` daemon process listens for requests for the specified service at each socket. When it receives a request, the `inetd` daemon process invokes the appropriate daemon process to handle the request.

The BOOTP and TFTP services must have their entries activated in the `inetd.conf` file in order to invoke their respective daemons (by default, their entries are commented out and must be activated). If you are designing your own application and have added services to the `inetd.conf` database file, define these services in the `services` database file. The `services` database file **must** list the services that you want the `inetd` daemon process to invoke.

To enable TCP wrappers functionality in a service that the `inetd` daemon process starts, the `inetd` daemon process must first invoke the `tcpd` daemon to start the service rather than the `inetd` daemon process invoking the service directly. When the `tcpd` daemon invokes a service, it checks the `hosts.allow` and `hosts.deny` files to determine if the client has access to the requested service. You can configure the services BOOTP and TFTP in the `inetd.conf` file so that the `inetd` daemon process invokes the `tcpd` daemon first, rather than invoking the requested service directly. The `tcpd` daemon can then check the `hosts.allow` and `hosts.deny` files for the client's access to the requested service.

#### NOTE \_\_\_\_\_

The STCP `inetd.conf` database file does not contain entries for FTP or TELNET; their daemon processes are

started individually, not as child processes of the `inetd` daemon process.

Within each entry for a service, specify a type of protocol (for example, `udp`). Each protocol specified in the `inetd.conf` file must have an entry in the file `(master_disk)>system>stcp>protocols`.

The template `inetd.conf` file is in the directory `(master_disk)>system>stcp>templates`. In a copy of this template file, enter each service entry on a single line according to the template format. When you have finished creating your own `inetd.conf` database file, save the copy in the `(master_disk)>system>stcp` directory. STCP must be able to find the `inetd.conf` file in this directory.

The template file includes “trivial” services such as `echo`, `discard`, `daytime` (human-readable time), `time` (machine-readable time, in the form of number of seconds since midnight, January 1, 1900) and `chargen` (character generator). These services are TCP- and UDP-based. They are not user services; they are available only through `inetd`.

After you modify the `inetd.conf` and `xinetd.conf` database files, you must stop and restart the `inetd` and `xinetd` daemon processes to activate the changes, or you can activate those changes immediately by issuing the following `kill` command.

```
kill -s HUP pid
```

For the `pid` argument, specify the process ID of the `inetd` daemon process. To determine the process ID of the `inetd` and `xinetd` daemon processes, issue the command `lui -process_id`. The command output lists the process ID of all processes.

## How to Edit the `inetd.conf` File

Each service entry in the `inetd.conf` file contains up to seven fields. Provide a value for each of the first six fields and optionally for the seventh field. Separate each field with any number of blank spaces or horizontal tabulation (tab) characters. To create a comment line, begin the line with the number sign (`#`); routines that search the file ignore all characters from the `#` character to the end of the line.

[Table 5-6](#) describes the fields that appear in an entry in the `inetd.conf` file.

**Table 5-6. Fields in the `inetd.conf` File**

Field	Description
<code>service name</code>	Identifies the name of the service, as listed in the <code>services</code> file. Required.
<code>socket type</code>	Identifies the appropriate socket type: <code>stream</code> for a TCP virtual circuit protocol service or <code>dgram</code> for a UDP datagram protocol service. Required.
<code>protocol</code>	Identifies the name of the protocol associated with the service (for example, <code>udp</code> or <code>tcp</code> ). The protocol must be listed in the <code>protocols</code> file. Required.
<code>wait or nowait (mode)</code>	Identifies the mode associated with the service. Specify the value <code>wait</code> if the <code>inetd</code> daemon process is to suspend listening only on the port until the invoked (single-threaded) server has finished executing. Specify the value <code>wait</code> for datagram sockets. Specify the value <code>nowait</code> if the <code>inetd</code> daemon process is to continue listening on the port after invoking the server (a multithreaded server). Typically, you specify the value <code>nowait</code> for stream sockets. Required.
<code>user</code>	Identifies the user name and group name for the daemon process, enabling you to control the privileges with which the daemon process runs. The <code>user</code> value has the format <code>user_name.group_name</code> . The default <code>user</code> is <code>root.root</code> . However, for security, you may want the user <code>nobody.SysAdmin</code> to run some daemon processes. Required.
<code>server program</code>	Identifies either the path name of the server program (daemon process) that you want the <code>inetd</code> daemon process to invoke in order to perform the requested service, or the value <code>internal</code> if <code>inetd</code> itself provides the service internally. Required.
<code>server program arguments</code>	Identifies the arguments to the server program specified in the <code>server program</code> field. If the specified server program must be invoked with command-line arguments, specify those arguments in this field. Do <b>not</b> enter the name of the server. Specify the value <code>internal</code> in this field instead of any arguments if <code>inetd</code> itself provides the service internally. Optional.

To enable the `inetd` daemon to first invoke the `tcpd` daemon to start the service, you must modify the service's entry in the `inetd.conf` file that you create from the `inetd.conf` file.

This section provides the following examples.

- [“Configuring the `tcpd` Daemon to Invoke the `swat` Daemon” on page 5-24](#)
- [“Configuring the `tcpd` Daemon to Invoke the `tftpd` and `bootpd` Daemons” on page 5-24](#)

## Configuring the `tcpd` Daemon to Invoke the `swat` Daemon

The `swat` daemon, an administrative tool of the SAMBA file server, provides an example of using the `inetd.conf` file to first invoke the `tcpd` daemon for access checking.

The following entry does not configure TCP wrappers functionality; this entry enables the `inetd` daemon process to start a service by directly invoking the daemon for the service.

```
swat    stream    tcp    nowait    root    >system>samba>command_library>swat.pm
```

The following entry configures TCP wrappers functionality; it invokes the `tcpd` daemon first, and the `tcpd` daemon then invokes the `swat` daemon. Note that this entry is **one line**, and that the ampersand and plus characters (`&+`) indicate a continuation line.

```
swat    stream    tcp    nowait    root    >system>stcp>command_library>tcpd.pm &+
>system>samba>command_library>swat.pm
```

These entries exist as comments in the `inetd.conf` file. To activate a comment, delete the number sign (`#`) and space at the beginning of the line.

## Configuring the `tcpd` Daemon to Invoke the `tftpd` and `bootpd` Daemons

The `tftpd` and `bootpd` daemons also provide examples of using the `inetd.conf` file to first invoke the `tcpd` daemon for access checking.

The following lines are examples of entries in the `inetd.conf` file that do not configure TCP wrappers functionality.

```
tftp dgram  udp  wait  root  >system>stcp>command_library>tftpd.pm
bootp dgram  udp  wait  root  >system>stcp>command_library>bootpd.pm
```

The following entries configure TCP wrappers functionality; these entries configure the `tcpd` daemon to invoke the daemons `tftpd` and `bootpd`.

```
tftp dgram  udp  wait  root  >system>stcp>command_library>tcpd.pm >system>stcp>&+
command_library>tftpd.pm
bootp dgram  udp  wait  root  >system>stcp>command_library>tcpd.pm >system>stcp>&+
command_library>bootpd.pm
```

These entries exist as a comment in the `inetd.conf` file. To activate these entries, delete the number sign (`#`) and space at the beginning of each line.

## Sample `inetd.conf` File

Figure 5-5 shows an excerpt from the template `inetd.conf` file. Note that this excerpt contains entries that enable the user `root` to invoke the trivial services ECHO, DISCARD, DAYTIME, CHARGEN, and TIME. The file also contains, as comments,

lines to invoke the `bootpd` and `tftpd` daemons, and the `swat` administrative tool with or without TCP wrappers functionality.

```
# The parameters are in this order:
# service-name
# endpoint-type protocol
# wait-status
# uid
# server-program (full pathname)
# server-arguments
#
echo    stream tcp nowait root    internal  internal
echo    dgram  udp wait  root    internal  internal
discard stream tcp nowait root    internal  internal
discard dgram  udp wait  root    internal  internal
daytime stream tcp nowait root    internal  internal
daytime dgram  udp wait  root    internal  internal
chargen stream tcp nowait root    internal  internal
chargen dgram  udp wait  root    internal  internal
time    stream tcp nowait root    internal  internal
time    dgram  udp wait  root    internal  internal
#
# Uncomment the following lines only if you want to use the bootp or tftpd server.
# You should be very careful since there are security issues with using tftp.
# tftp dgram  udp wait root >system>stcp>command_library>tftpd.pm
# bootp dgram  udp wait root >system>stcp>command_library>bootpd.pm
#
# If you are using tcpwrappers to control access to tftp, use the following:
# tftp dgram  udp wait  root >system>stcp>command_library>tcpd.pm >system>s
+tcp>command_library>tftpd.pm
#
# The following sample lines apply to Samba 2.1 for VOS.
#
# Use the following line if you are using the Samba Web Administration Tool (SWAT):
# swat stream tcp nowait root >system>samba>command_library>swat.pm
#
# Use the following line if you are using tcpwrappers to control access to swat:
# swat stream tcp nowait root >system>stcp>command_library>tcpd.pm
+>system>samba>command_library>swat.pm
#
# The following sample lines apply to Samba 3.0 for VOS.
#
# Use the following line if you are using the Samba Web Administration Tool (SWAT):
# swat stream tcp nowait root >system>samba>sbin>swat.pm
#
# Use the following line if you are using tcpwrappers to control access to swat:
# swat stream tcp nowait root >system>stcp>command_library>tcpd.pm
+>system>samba>sbin>swat.pm
```

**Figure 5-5. Excerpt from the Template `inetd.conf` File**

## Related Information

For a description of the `inetd` and `tcpd` daemon process, see [Chapter 8](#). The `hosts.allow` and `hosts.deny` database files are described in “[The hosts.allow and hosts.deny Files and TCP Wrappers](#)” on page 5-10. The `services` database file is described in “[The services File and Service Information](#)” on page 5-46. For information about the SAMBA file server, see the current software release bulletin (SRB) for Samba. For more information about the `kill` command, see the *OpenVOS POSIX.1 Reference Guide* (R502).

## The networks File and Network Information

The `networks` database file contains information about the networks in your STCP configuration. The following sections provide additional information about the `networks` file.

- “[How the networks File Is Used](#)” on page 5-26
- “[How to Edit the networks File](#)” on page 5-26
- “[Sample networks File](#)” on page 5-27

### How the networks File Is Used

STCP checks the `networks` file when it needs information about the available networks (the name, the network number of the IPv4 address, and the alias of each network to which you want to connect using STCP). Also provide information about unofficial aliases and subnetworks so that the `networks` file is complete and accurate for your system. (See “[Obtaining IP Addresses and Domain Names](#)” on page A-22 for information about obtaining IP network numbers.)

To specify the networks in your configuration, create your own copy of the template file `(master_disk)>system>stcp>templates>networks`. Place the final version of your `networks` file in the `(master_disk)>system>stcp` directory.

### How to Edit the networks File

In your `networks` file, create an entry for each network on a single line. Separate the fields of a `networks` file entry with any number of blank spaces or horizontal tabulation (tab) characters. Use a carriage return to separate lines. To create a comment line, begin the line with the number sign (`#`); routines that search the file ignore all characters from the `#` character to the end of the line. [Table 5-7](#) describes the fields used in each entry.



**Table 5-7. Fields in the *networks* File**

Field	Description
network name	Identifies the official network name associated with the network. A network name defined in the <i>networks</i> file is a text string of up to 64 characters that can contain the letters of the alphabet (A through Z and a through z), the numbers 0 through 9, the hyphen (-), and the period (.). The first character of the text string must be alphanumeric. (This definition of a network name is derived from RFC 952 and RFC 1123.) Stratus recommends that network names not exceed 11 characters. Required.
network number	Identifies the network number of the IP address. Use standard dot-notation form when specifying the network number of the IP address. Required.
alias	Identifies an alias (another name that can be used to identify the network). You can specify more than one alias, but to avoid confusion, do not create many aliases. Optional.
comment	Identifies a comment for the entry. Optional.

## Sample *networks* File

Figure 5-6 shows the template *networks* file which uses the standard template network numbers (do not assign the Ethernet numbers to your network).

```
# Sample Networks
#       Contains information on networks connected to this machine.
#
# The fields are:
# official network name  network number alias  [ alias ]  [ alias ]...
#
# For example:
#      class-A-net        89                ether eth net      # 89.0.0.0
#      class-B-net        90.3              # 90.3.0.0
#      class-C-net        91.66.77          # 91.66.77.0
#
# network name  network number  alias [ alias ] [ alias ]...
#
# Standard VOS Entries
#
loopback        127                local                # 127.0.0.0
maintnet        10.10.1            # 10.10.1.0
syncnet        10.20.1            # 10.20.1.0
#
# Local networks
#
ethernet        89                ether eth net
```

**Figure 5-6. Template *networks* File**

## The `nsswitch.conf` File and Name Server Information

The `nsswitch.conf` file specifies the name services and their search order, which a system uses for lookups that use the name server switch (nsswitch) facility. Though not a database file, you need to configure this file when you configure relevant database files. This file is used only by programs that have been built under POSIX rules in OpenVOS Release 17.1 or later. The `nsswitch.conf` file supports both IPv4 and IPv6 protocols.

The following sections provide additional information about the `nsswitch.conf` file.

- [“How the `nsswitch.conf` File Is Used” on page 5-28](#)
- [“How to Edit the `nsswitch.conf` File” on page 5-28](#)
- [“Sample `nsswitch.conf` file” on page 5-29](#)
- [“Related Information” on page 5-31](#)

### How the `nsswitch.conf` File Is Used

The `nsswitch.conf` file lists databases that are sources of information about IP addresses. By modifying the list of databases, you can customize how STCP performs various Internet lookup functions such as name resolution.

Each entry in the list consists of a database name and one or more sources. If a single source is listed, STCP terminates the search if the information is not found. If two or more sources are listed, STCP searches the first listed source, and if the information is not found, STCP continues the search with the next listed source. For example, the entry `hosts: files dns` specifies that STCP searches the local `hosts` file first. If no match is found, STCP then searches the Domain Name Service (DNS).

The configuration defined in the `nsswitch.conf` file is captured the first time an application performs a lookup and is reloaded on subsequent lookups if the modification date on the file has been changed to a more recent date.

The template `nsswitch.conf` file is located in the `(master_disk)>system>stcp>templates` directory. In a copy of this file, enter information according to the template format. When you have finished editing the copy, save it in the `(master_disk)>system>stcp` directory.

### How to Edit the `nsswitch.conf` File

In the `nsswitch.conf` file, each entry contains a database name followed by a colon and a space (`:` ), and then one or more sources, listed in search order, as follows:

```
database: source [source ... ]
```

Table 5-8 shows the `database` values for supported databases, and Table 5-9 shows the `source` values.

**Table 5-8. `database` Values in the `nsswitch.conf` File**

<i>database</i> Values	Description
<code>hosts</code>	The hosts database file. See “ <a href="#">The hosts File and Host Name Information</a> ” on page 5-7.
<code>networks</code>	The networks database file. See “ <a href="#">The networks File and Network Information</a> ” on page 5-26.
<code>services</code>	The services database file. See “ <a href="#">The services File and Service Information</a> ” on page 5-46.
<code>services_compat</code>	An alternative method for obtaining information that STCP uses when <code>compat</code> is the source value for <code>services</code> . Do not change the contents of this file.
<code>protocols</code>	The protocols database file. See “ <a href="#">The protocols File and Protocol Information</a> ” on page 5-40.

**Table 5-9. `source` Values in the `nsswitch.conf` File**

<i>source</i> Values	Description
<code>compat</code>	Use the <code>database_compat</code> database.
<code>nis</code>	Search the Network Information Service.
<code>files</code>	Search local files.
<code>dns</code>	Search the Domain Name Service.

## Sample `nsswitch.conf` file

Figure 5-7 shows the template STCP `nsswitch.conf` file.

```
# Sample nsswitch.conf(5) - name service switch configuration file
#
# This configuration file is only used by programs that have been built
# under POSIX rules in release 17.1 or later. If this configuration file
# is not present, host name lookups will be compatible with the older
# versions of the DNS software. To avoid confusion, we recommend that you
# defer using this configuration file until all applications have been
# rebuilt using POSIX
```

(Continued on next page)

```
#
# The non-POSIX DNS software (and all the DNS software prior to release
# 17.1) resolved hosts by first using DNS and then using the
# >system>stcp>hosts file.  This has a couple of minor problems:
#
# 1.  If there is a bad entry in the hosts file, it won't get detected
#      until something causes DNS to go away.  This could result in
#      confusion exactly when one doesn't want it.
#
# 2.  There is no way to over-ride a bad host coming back from DNS.
#
# 3.  Most other implementations search the hosts file first.
#
# None of these are compelling enough to make an incompatible change
# to the default behavior; however, you can switch the search order
# using this file to avoid the above issues.  To search the hosts file
# first, you want:
#
#             hosts: files dns
#
# to do DNS first and remain 100% compatible with the behavior of
# previous releases of VOS, you want:
#
#             hosts: dns files
#
# Also, note that the non-POSIX and pre 17.1 software never used dns for
# looking up networks.  The new software does.  By default, it does this
# after looking in the >system>stcp>networks file.  One can configure this
# behavior, too.  There usually isn't much useful information that comes
# back from DNS for networks.  If you want to turn off DNS lookups for
# networks, do this:
#
#             networks: files
#
group: compat
group_compat: nis
hosts: files dns
networks: files dns
passwd: compat
passwd_compat: nis
shells: files
services: compat
services_compat: nis
protocols: files
rpc: files
```

**Figure 5-7. Template *nsswitch.conf* File**

## Related Information

For more information about DNS, see [“Domain Name Service” on page A-23](#).

## The *ospfdconf* File and Routing Information

The *ospfdconf* file specifies local routing information for the *ospfd* daemon process. The following sections provide additional information about the *ospfdconf* file.

- [“How the \*ospfdconf\* File Is Used” on page 5-31](#)
- [“How to Edit the \*ospfdconf\* File” on page 5-32](#)
- [“Sample \*ospfdconf\* Files” on page 5-37](#)
- [“Related Information” on page 5-40](#)

Before you read this section, review the information in [“OSPF Routing” on page A-19](#).

## How the *ospfdconf* File Is Used

The *ospfdconf* file specifies routing information for an OpenVOS module that enables it to participate in an Open Shortest Path First (OSPF) autonomous system (AS).

The template *ospfdconf* file resides in the directory (master\_disk)>system>stcp>templates. In a copy of this template file, enter information according to the template format. When you have finished creating your own version of the *ospfdconf* file, save the version in the directory (master\_disk)>system>stcp. By default, STCP uses the *ospfdconf* file in this directory.

Initially, define the following in the *ospfdconf* file:

- router ID, which identifies the OpenVOS module in the OSPF AS
- area or areas to which network interfaces on the module belong and various attributes of each membership, including the IPv4 addresses in and the authentication key for each area
- interfaces that participate in the area

To add a new interface as a member of the OSPF AS, update the *ospfdconf* file and then stop and restart the *ospfd* daemon.

## How to Edit the `ospfdconf` File

The `ospfdconf` file contains the following sections:

- router ID, in an `RTRID` entry
- area descriptions that each contain `AREA`, `RANGE`, and `AUTH` entries
- list of interfaces in multiple `IF` entries

Optionally, include sections that define point-to-point routes, in `HOST` entries and logical links to the backbone, in `VIRTUAL` entries.

Separate each field in an entry with any number of blank spaces or horizontal tabulation (tab) characters. To create a comment line, begin the line with the number sign (#); routines that search the file ignore all characters from the # character to the end of the line.

Table 5-10 describes the fields used in each entry in the `ospfdconf` file.

**Table 5-10. Entries in the `ospfdconf` File** (Page 1 of 5)

Keyword	Description
RTRID	<p>Identifies a router ID. Required.</p> <p>Each module that has network interfaces that participate in one or more OSPF areas has a single router ID. Use the following format in <code>RTRID</code> entries:</p> <pre>RTRID: router_id 0</pre> <p>The first value, <code>router_id</code>, specifies a 32-bit number, in dotted decimal notation, that uniquely identifies the module within its AS. Use the IPv4 address of one of the interfaces as a router ID to ensure a unique value.</p> <p>The second value, <code>0</code>, indicates that the module is not a boundary router and will not advertise external routes. Because OpenVOS modules support only the OSPF routing protocol, an OpenVOS module cannot be a boundary router and this value must always be <code>0</code>.</p> <p>The following entry sets the router ID to 10.111.57.0 and specifies that the module is not a boundary router.</p> <pre>RTRID: 10.111.57.0 0</pre>

**Table 5-10. Entries in the *ospfdconf* File** (Page 2 of 5)

Keyword	Description
AREA	<p>Identifies the area or areas to which the interfaces belong. Required.</p> <p>Each AREA entry specifies an area ID and whether the area is a stub area. At least one AREA entry is required, but create multiple AREA entries to specify membership in multiple areas. For each area, also create one or more RANGE entries, an AUTH entry, and one or more IF entries. Use the following format in an AREA entry:</p> <pre>AREA: area_id {0   1} metric</pre> <p><b>Note:</b> If you create multiple AREA entries, you must configure one of the interfaces to participate in the backbone area.</p> <p>The <i>area_id</i> field specifies a 32-bit number in dotted decimal notation that identifies a collection of contiguous networks and hosts. If subnetted networks are each a separate area, you can use the IP network number for the area ID. The area ID for the backbone area is 0.0.0.0.</p> <p>The second field specifies whether the area is a transit area or stub area. A value of 0 specifies a transit area, and a value of 1 specifies a stub area.</p> <p>The <i>metric</i> field indicates the cost of the default summary link that a border router advertises. Enter 0 in this field if the area is a transit area, or enter another value in this field if the area is a stub area.</p> <p>The following entry sets the area ID to 0.0.0.1, specifies that the area is a transit area, and specifies 0 in the <i>metric</i> field.</p> <pre>AREA: 0.0.0.1 0 0</pre>
RANGE	<p>Specifies the IPv4 addresses that comprise an area. Create at least one RANGE entry for each AREA entry, except for the backbone area. Use the following format:</p> <pre>RANGE IPaddr subnet_mask</pre> <p>The <i>IPaddr</i> field generally specifies a subnetwork number that identifies the area. The <i>subnet_mask</i> field specifies the subnet mask used by the subnetwork.</p> <p>If the area comprises several subnetworks, create multiple RANGE entries, one for each subnetwork.</p> <p>The following entry specifies that the area consists of all IPv4 addresses in the 192.168.57.0 subnet and that the subnet uses a subnet mask of 255.255.255.0:</p> <pre>RANGE: 192.168.57.0 255.255.255.0</pre>

**Table 5-10. Entries in the *ospfdconf* File** (Page 3 of 5)

Keyword	Description	
AUTH	<p>Specifies the authentication type for the area. Create one AUTH entry for each AREA entry. Use the following format:</p> <pre>AUTH: {0   1}</pre> <p>A value of 0 turns off authentication, and a value of 1 turns on authentication by password. Configure the password in the AUTH_KEY value of the IF entry for each interface.</p> <p>The following entry specifies that the area uses authentication by password.</p> <pre>AUTH: 1</pre>	
IF	<p>Specifies the configuration of any interfaces on the module that belong to an area in the following format:</p> <pre>IF: area_id IPaddr type metric retransmit_interval transmit_delay priority hello_interval router_dead_interval auth_key</pre> <p>The configuration of an interface consists of values for the following fields:</p>	
	<i>area_id</i>	A 32-bit number in dotted decimal notation that indicates the area to which the interface belongs.
	<i>IPaddr</i>	A 32-bit number in dotted decimal notation that indicates the IPv4 address of the interface.
	<i>type</i>	The value 1 indicates that the interface is a broadcast type; the value 2 indicates a nonbroadcast multiaccess (NBMA) type; and the value 3 indicates a point-to-point type. 2, NBMA, is not supported on OpenVOS modules.
	<i>metric</i>	A value that indicates the cost of using the interface. This value is advertised in link state advertisements (LSA). Values 1 through 65535 are valid.
	<i>retransmit_interval</i>	A value that indicates the number of seconds between LSA and database description packets, and retransmissions of link state request packets from this interface.
	<i>transmit_delay</i>	A value that indicates the estimated number of seconds it takes to transmit an LSA over the interface. Specify a value greater than 0; when calculating the value, include transmission and propagation delays.



**Table 5-10. Entries in the *ospfdconf* File** (Page 4 of 5)

Keyword	Description	
IF (Cont.)	<i>priority</i>	<p>A value that routers in broadcast and NBMA networks use to determine which routers become the designated router and backup designated router for the area. Values 0 through 256 are valid.</p> <p>During a designated router or backup designated router election process, the router with the highest priority becomes the DR or BDR. A large value assigns a high priority to the router. Assign the value 0 to the interface to prevent the router from becoming either a designated router or backup designated router.</p>
	<i>hello_interval</i>	A value that indicates the number of seconds between Hello packets sent on this interface. Default value: 10 seconds.
	<i>router_dead_interval</i>	A value that specifies the number of seconds between Hello packets from the interface before the router's neighbors declare it down. Specify a value several times that of the <i>hello_interval</i> to account for occasional delays or losses of Hello packets. Default value: 40 seconds.
	<i>auth_key</i>	A password of up to 8 characters shared by the neighboring routers. If you specify an <i>auth_key</i> , also specify the value 1 in the AUTH entry for the area. All routers in an area must share the same authentication type and password.
	<p>The following entry configures an interface with an IPv4 address of 192.168.57.1 in area 0.0.0.2:</p> <pre>IF: 0.0.0.2 192.168.57.1 1 5 20 2 0 30 120 passwd</pre> <p>In this example, 1 specifies that the interface is to a broadcast network, and the remaining values specify a cost of 5, a retransmit interval of 20 seconds, a transmit delay of 2 seconds, a priority of 0, a hello interval of 30 seconds, a router-dead interval of 120 seconds, and a password of passwd.</p>	

**Table 5-10. Entries in the *ospfdconf* File** (Page 5 of 5)

Keyword	Description
VIRTUAL	<p>Corresponding entries of this type on two border routers logically connect an area to the backbone. The two border routers must each have an interface to a common area that is not part of the backbone.</p> <p>To create a logical link between two routers or modules, create a VIRTUAL entry in the <i>ospfdconf</i> file on each. Use the following format for each entry:</p> <pre>VIRTUAL: neighbor area_id IF_values...</pre> <p>The <i>neighbor</i> field specifies the router ID of the module or router that forms the other end of the virtual link. It must be connected to the backbone, either logically or physically.</p> <p>The <i>area_id</i> field specifies the ID of the <i>transit area</i>, the area shared by the two interfaces.</p> <p>The <i>IF_values</i> fields consist of the following standard fields for an interface specification: <i>transmit_delay</i>, <i>retransmit_interval</i>, <i>hello_interval</i>, <i>router_dead_interval</i>, and <i>auth_key</i>. These values are described for the IF entry.</p> <p>The following entry configures a virtual link to a border router whose router ID is 10.112.0.1, through area 0.0.0.2.</p> <pre>Virtual: 10.112.0.1 0.0.0.2 2 20 30 120 passwd</pre> <p>This example specifies a delay of 2 seconds, a retransmit interval of 20 seconds, a hello interval of 30 seconds, a router-dead interval of 120, and a password of passwd.</p>
HOST	<p>Configures a host route for a point-to-point network, looped router interface, or an IP host directly connected to an interface on the module. Use the following format:</p> <pre>Host: area_id IPaddr metric</pre> <p>The <i>area_id</i> field specifies a unique ID for the area. Only the interface specified in this configuration and the interface on the local router can participate in this area.</p> <p>The <i>IPaddr</i> field specifies the IPv4 address of the point-to-point network, looped router interface, or IP host.</p> <p>The <i>metric</i> field indicates the cost of using the interface.</p> <p>The following entry configures a route to a directly attached interface that has the IPv4 address 10.111.103.55, specifies an area ID of 0.0.0.5, and a cost of 1 for using the interface.</p> <pre>Host: 0.0.0.5 10.111.103.55 1</pre>

## Sample *ospfdconf* Files

[Figure 5-8](#) shows a sample *ospfdconf* file for a border router that is physically connected to the backbone.

```
# ABR Router
RTRID: 172.16.10.55 0

# Area 0.0.0.0 (backbone)
AREA: 0.0.0.0 0 0
AUTH: 0

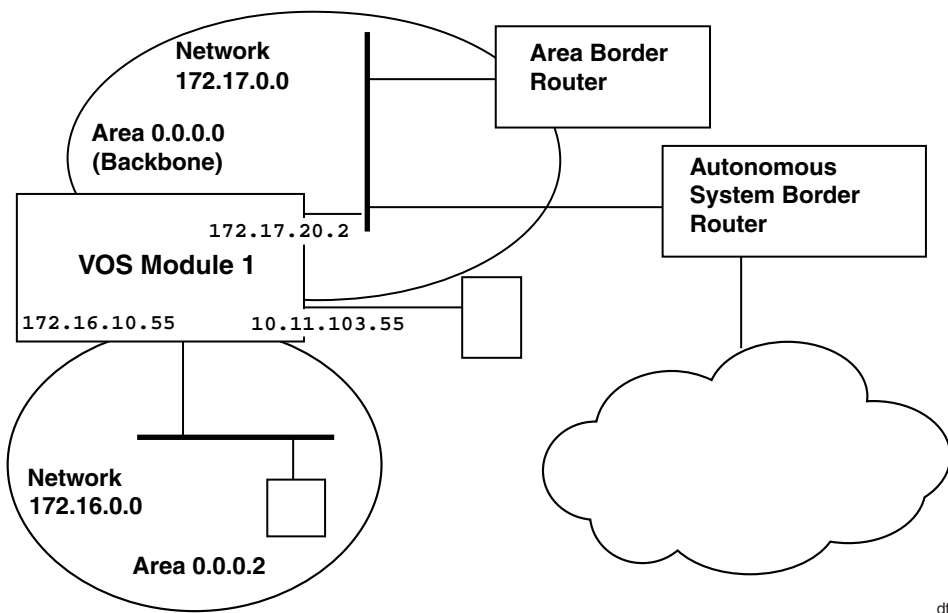
# Area 0.0.0.2
AREA: 0.0.0.2 0 0
RANGE: 172.16.10.0 255.255.255.0
AUTH: 0

# Broadcast Interface (to backbone)
IF: 0.0.0.0 172.17.20.2 1 1 20 2 2 10 40 password

# Broadcast Interface (to area 0.0.0.2)
IF: 0.0.0.2 172.16.10.55 1 1 20 2 2 10 40 password
# Host Route
HOST: 0.0.0.2 10.111.103.55 10
```

**Figure 5-8. Sample *ospfdconf* File: Physical Connection to the Backbone**

[Figure 5-9](#) shows an OpenVOS module that is a border router physically attached to the backbone; OpenVOS Module 1 contains an *ospfdconf* file similar to the one in [Figure 5-8](#).



**Figure 5-9. System Configuration: Physical Connection to the Backbone**

Figure 5-10 shows a sample *ospfdconf* file for a border router (OpenVOS Module 2) that is logically connected to the backbone.

**Figure 5-10. Sample *ospfdconf* File: Logical Connection to the Backbone**

```
# ABR Router
RTRID: 192.168.6.44 0

# Area 0.0.0.0 (backbone)
AREA: 0.0.0.0 0 0
AUTH: 1

# Virtual Link
VIRTUAL: 192.168.5.1 0.0.0.2 2 20 30 120 password
```

*(Continued on next page)*

**Figure 5-10. (Continued)**

```
# Area 0.0.0.2
AREA: 0.0.0.2 0 0
RANGE: 192.168.5.0 255.255.255.0
RANGE: 192.168.6.0 255.255.255.0
AUTH: 1

# Area 0.0.0.3
AREA: 0.0.0.2 0 0
RANGE: 192.168.3.0 255.255.255.0
AUTH: 1

# Broadcast Interface (to area 0.0.0.3)
IF: 0.0.0.0 192.168.3.5 1 1 20 2 2 10 40 password

# Broadcast Interface (to area 0.0.0.2)
IF: 0.0.0.2 192.168.6.44 1 1 20 2 2 10 40 password
```

[Figure 5-11](#) shows two OpenVOS modules. OpenVOS Module 1 is a border router physically attached to the backbone. OpenVOS Module 2 is an area border router logically attached to the backbone by a virtual link to OpenVOS Module 1. OpenVOS Module 2 contains an *ospfdconf* file similar to the one in [Figure 5-10](#).

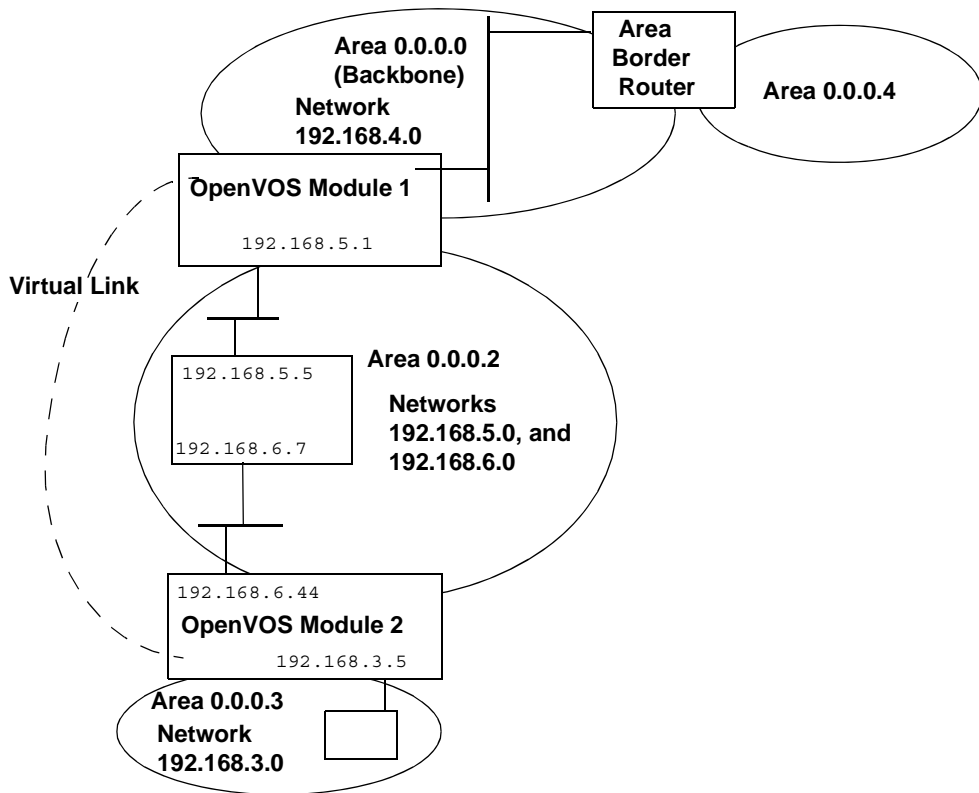


Figure 5-11. System Configuration: Logical Connection to the Backbone

## Related Information

[“OSPF Routing” on page A-19](#) provides an overview of the Open Shortest Path First (OSPF) routing protocol.

## The protocols File and Protocol Information

The `protocols` database file contains the Internet (IP) protocols used by applications running over STCP on your system. The following sections provide additional information about the `protocols` file.

- [“How the protocols File Is Used” on page 5-41](#)
- [“How to Edit the protocols File” on page 5-41](#)
- [“Sample protocols file” on page 5-42](#)

## How the protocols File Is Used

Only protocols defined in the `protocols` file can be used by applications running over STCP.

The template `protocols` database file, located in the directory `(master_disk)>system>stcp>templates`, has entries for the protocols IP, ICMP, TCP, and UDP. Typically, you simply copy the template file and save it in the `(master_disk)>system>stcp` directory.

To reference a protocol not supplied with the STCP software, you must create an entry for that protocol in your copy of the `protocols` file.

## How to Edit the protocols File

Each entry in the `protocols` file requires a single line with the following information:

```
protocol number alias comment
```

Separate the fields of a `protocols` file entry with any number of blank spaces or horizontal tabulation (tab) characters. Use a carriage return to separate lines. To create a comment line, begin the line with the number sign (#); routines that search the file ignore all characters from the # character to the end of the line.

[Table 5-11](#) describes the fields used in each entry.

**Table 5-11. Fields in the protocols File**

Field	Description
<code>protocol</code>	Identifies the official protocol name associated with the protocol. The protocol name is a text string of up to 64 characters that contains the letters of the alphabet (A through Z and a through z), the numbers 0 through 9, the hyphen (-), and the period (.). The first character of the text string must be alphanumeric. This definition of a protocol name is derived from RFC 952 and RFC 1123.
<code>number</code>	Identifies the protocol number. Required.
<code>alias</code>	Identifies an alias (another name that can be used to identify the protocol). You can specify more than one alias, but to avoid confusion, you should not create many aliases. Optional.
<code>comment</code>	Identifies a comment for the entry. Optional.

## Sample protocols file

Figure 5-12 shows the template protocols file.

```
# Internet (IP) Protocols
# protocol number alias(es) # comment(s)
#
ip          0      IP      # internet pseudo protocol number
icmp       1      ICMP    # internet control message protocol
tcp        6      TCP     # transmission control protocol
udp       17      UDP     # user datagram protocol
```

**Figure 5-12. Template protocols File**

## The `resolv.conf` File and Name Server Information

The `resolv.conf` database file contains information that is read by the name resolver routines, a set of routines that provide access to the Internet Domain Name System (DNS).

The following sections provide additional information about the `resolv.conf` file.

- [“How the `resolv.conf` File Is Used” on page 5-42](#)
- [“How to Edit the `resolv.conf` File” on page 5-43](#)
- [“Sample `resolv.conf` file” on page 5-46](#)
- [“Related Information” on page 5-46](#)

### How the `resolv.conf` File Is Used

The resolver routines read information in the `resolv.conf` file (if the file exists) when they are first invoked by a process. If your network configuration relies on a remote name server to resolve symbolic host names, use `resolv.conf` to identify the name servers that the resolver routines should query.

The template `resolv.conf` file is located in the `(master_disk)>system>stcp>templates` directory. In a copy of this file, enter information according to the template format. When you have finished editing the copy, save it in the `(master_disk)>system>stcp` directory.

In addition, you may want to maintain a customized `hosts` file to ensure that STCP can resolve host names even if a name server is unavailable. The STCP resolver routines will try to reach the name server before checking the `hosts` file for information.

When a name server is available to resolve host names and a domain name is specified, the domain name is appended to the host name. If a search list is specified,



the domain names in the search list are individually appended to the host name in the order that they appear in the search list until the host is found. If the `hosts` file is used to resolve a host name, the domain name is not appended to the host name.

#### NOTE

Because OpenVOS does **not** currently support name server software, do not identify your local host as a name server.

## How to Edit the `resolv.conf` File

In the `resolv.conf` file, each entry has a keyword field and a value field, which together create a keyword-value pair. The keyword and value in a keyword-value pair must appear on a single line, and the keyword must start the line. The value follows the keyword and must be separated from the keyword by a space.

[Table 5-12](#) shows the possible values for the keyword and value fields. The `domain` and `search` keywords are mutually exclusive. If more than one instance of the `domain` and/or `search` keywords appears in this file, the last instance will override previous instances. The `options` and `sortlist` fields are for POSIX-compliant applications and do not appear in the template file. Applications that are not POSIX-compliant ignore these fields.

**Table 5-12. Fields in the `resolv.conf` File** (Page 1 of 2)

keyword Field Values	value Field Values	Description
<code>domain</code>	<code>name</code>	Specifies your domain name, a subset of Internet members that share common services, such as name and address translation. For example, most commercial members of the Internet are assigned to the domain named <code>com</code> . Since you can have only one domain, you can have only one domain entry.
<code>nameserver</code>	<code>address</code>	Specifies the IP address of the name server that you want the resolver routines to query. Specify up to three name servers. The resolver routines query the name servers in the order in which they are specified.
<code>options</code>	<code>options</code>	Specifies various options. These options may also be specified in the <code>RES_OPTIONS</code> environment variable. <a href="#">Table 5-13</a> lists and describes the values for <code>options</code> .

**Table 5-12. Fields in the *resolv.conf* File** (Page 2 of 2)

keyword Field Values	value Field Values	Description
search	<i>name1 name2 name3 ... name6</i>	<p>Specifies a search list of up to six domain names for host-name lookup. Your domain name normally determines the search list. To change the search process, list the domain names in the search list after the <i>search</i> keyword, with spaces or tabs separating the names. Resolver queries will be attempted using each component of the search list in turn until a match is found. The search list is limited to six domains with a total of 256 characters. Note that the search process will generate considerable network traffic if the servers for the listed domains are not local, and queries will time out if no server is available for one of the domains.</p> <p>The following example shows a sample search list with three domains.</p> <pre>search sales.corp.com mktg.corp.com pubs.corp.com</pre>
sortlist	<i>IP/NETMASK</i>	<p>Defines, for POSIX compliance, the sort order for the results of <i>gethostbyname</i> and <i>getipnodebyname</i> functions. This list can specify up to ten <i>IP/NETMASK</i> pairs that partially determine the result ordering. If you do not define the <i>/NETMASK</i> value, STCP determines a default value based on the IP addresses using the standard rules for class A, B, or C IP addresses. You must specify v4 IP addresses, even on a system that supports v6 IP addresses. In cases where a <i>get</i> routine returns more than one IP address, the results are sorted as follows:</p> <ul style="list-style-type: none"> <li>• IP addresses that match those given in <i>sortlist</i> are returned first and in the order specified by <i>sortlist</i>.</li> <li>• If multiple returned IP addresses match the same entry in <i>sortlist</i>, the ordering from the name server is preserved.</li> <li>• IP addresses that do not match any in <i>sortlist</i> are returned last in the order provided by the name server.</li> </ul>

Table 5-13 lists and describes the values for the `options` field.

**Table 5-13. Values of the `options` Field in the `resolv.conf` File**

Value	Description
<code>ndots: number</code>	The minimum number of dots a name must have to be looked up in its <i>as-is</i> value; that is, without any appended domain or search suffixes. Valid values are 0 to 15; the default is 1. This option only affects whether the name is looked up as-is: If the as-is lookup fails, the resolver tries all of the suffixes specified by <code>search</code> (or the suffix specified by <code>domain</code> ) until it finds a match.
<code>timeout: seconds</code>	Sets the initial timeout for a DNS query. Repeated timeouts follow an exponential backoff algorithm. Valid values are 1 to 30 seconds; the default is 2.
<code>attempts: number</code>	Sets the number of attempts to query a particular DNS server. Valid values are 1 to 5; the default is 2.
<code>debug</code>	Enables a rather large volume of debugging output that is sent to <code>stdout</code> . Debugging output is most useful for debugging the resolver code, not applications that call it.
<code>no_tld_query</code>	Disables DNS queries on a name without dots.
<code>insecure1</code>	Disables checking the peer address to ensure that a response came from the DNS server that was queried. This check is an attempt to detect spoofed DNS responses. Do not use this option unless you have a compelling reason to use it.
<code>insecure2</code>	Disables checking that a DNS response is from the query that was sent. DNS servers include the query information in the response they send back; therefore, the client can, and should, check to ensure the response matches the query it sent. This check is an attempt to detect stale and/or spoofed DNS responses. Do not use this option unless you have a compelling reason to use it.
<code>rotate</code>	Rotates the name-server list after each query.

## Sample `resolv.conf` file

Figure 5-13 shows the template STCP `resolv.conf` file.

```
#           Sample resolv.conf file
#
# This is the resolver configuration file resolv.conf.

# Each entry is a keyword-value pair. There are three
# keyword-value configuration options, which are as follows:
# (note: keywords 'domain' and 'search' are mutually exclusive)
#
# nameserver      address
# domain          name
#or
# search          name1 name2 name3 name4 name5 name6
#
# keyword         value
#
nameserver        89.2.18.9
nameserver        89.2.1.10
domain            corp.com
# search sales.corp.com mktg.corp.com pubs.corp.com
```

**Figure 5-13. Template `resolv.conf` File**

## Related Information

For more information about DNS, see [“Domain Name Service” on page A-23](#).

## The *services* File and Service Information

The `services` database file contains the services available on your system.

The following sections provide additional information about the `services` file.

- [“How the `services` File Is Used” on page 5-47](#)
- [“How to Edit the `services` File” on page 5-47](#)
- [“Sample `services` File” on page 5-48](#)

### NOTE \_\_\_\_\_

RSN connections require an entry in the `services` file.  
For information about this entry, see the manual

## OpenVOS System Administration: Configuring a System (R287).

### How the `services` File Is Used

For STCP to recognize a service on the module, it must have an entry in the `services` file. This includes services such as TELNET and FTP, any services governed by the `inetd` daemon process, as well as any applications that you design or need to support (for example, additional services associated with TELNET incoming slave connections). (Services managed by the `inetd` daemon process must also have an entry in the `inetd.conf` file; local services managed by the `telnetd` daemon process must also have an entry in the `telnet.service` file.)

To define additional services, create an entry for each service in a copy of the template `services` file, located in the `(master_disk)>system>stcp>templates` directory. If you do not need to add any services, simply place a copy of this template file in the `(master_disk)>system>stcp` directory. When you have finished creating your own `services` file, save it in the `(master_disk)>system>stcp` directory.

### How to Edit the `services` File

Each entry in the `services` file requires a single line. Separate the fields of a `services` file entry with any number of blank spaces or horizontal tabulation (tab) characters. Use a carriage return to separate lines. To create a comment line, begin the line with the number sign (`#`); routines that search the file ignore all characters from the `#` character to the end of the line.

[Table 5-14](#) describes the fields used in each entry in the `services` file.

**Table 5-14. Fields in the `services` File** (Page 1 of 2)

Field	Description
Name	Identifies the official service name associated with the service. A service name defined in the <code>services</code> file is a text string of up to 64 characters that can contain the letters of the alphabet (A through Z and a through z), the numbers 0 through 9, the hyphen (-), and the period (.). The service name cannot contain spaces, tabs, newline characters, or number signs. The first character of the text string must be alphanumeric. (This definition of a service name is derived from RFC 952 and RFC 1123.) Required.
port/ protocol	Identifies the port number that remote users use to access this service, followed by the protocol name. The port number is separated from the protocol name by the slant (/) character. The <code>port/protocol</code> pair that you want to use must not be already assigned. Required.

Table 5-14. Fields in the services File (Page 2 of 2)

Field	Description
Service	Identifies an alias (another name that can be used to identify the service). You can specify more than one alias, but to avoid confusion, do not create many aliases. Optional.
comment	Identifies a comment for the entry. A comment is preceded by the number sign (#). Optional.

Sample services File

Figure 5-14 shows the template services file. Note that the two FTP service entries indicate that port 20 is the noninteractive FTP port and port 21 is the interactive FTP command port for the FTP server (ftpd).

Not all of the services listed in this file are necessarily applicable. The file simply lists various Internet-style services.

```
# Sample Network services, Internet style
#
#Name          port          Service
#
ftpdata        20/tcp
ftp            21/tcp          ftpd
telnet         23/tcp          telnetd
smtp           25/tcp
bootps         67/udp          bootpd
bootpc         68/udp          bootp
tftp           69/udp          tftpd
snmp           161/udp         snmpd
route          520/udp         router routed
echo           7/tcp
echo           7/udp
discard        9/tcp          sink null
discard        9/udp          sink null
daytime        13/tcp
daytime        13/udp
chargen        19/tcp          ttytst source
chargen        19/udp          ttytst source
time           37/tcp          timserver
time           37/udp          timserver
finger         79/tcp
ntalk          518/udp
```

(Continued on next page)

```
exec          512/tcp
domain        53/tcp          nameserver      # name-domain server
domain        53/udp          nameserver
nb_nmsrv      137/udp          netbios nameserver
nb_sssrv      139/tcp          netbios session server
nb_dgsrv      138/udp          netbios datagram server
```

**Figure 5-14. Template `services` File**

#### NOTE

Stratus strongly recommends that you use the standard service definition of `telnet` as the default TELNET login service associated with port 23.

## The *snmpconf* File and SNMP Information

The *snmpconf* database file is used by the *snmpd* daemon process, which supports the Simple Network Management Protocol (SNMP).

The following sections provide additional information about the *snmpconf* file.

- “SNMP Overview” on page 5-49
- “How the *snmpconf* File Is Used” on page 5-50
- “How to Edit the *snmpconf* File” on page 5-50
- “Sample *snmpconf* File” on page 5-51

## SNMP Overview

SNMP refers to a group of network management protocols and functions that rely on the IP stack to establish communications. SNMP uses the following software.

- The SNMP daemon (*snmpd*) runs on network equipment that is suitable for undergoing network management, such as host computers, routers, and concentrators. The daemon oversees the operation of the network equipment and maintains a database of network-related variables in a machine-readable file called a Management Information Base (MIB). STCP’s SNMP daemon is *snmpd* and supports Internet MIB-II (which holds standard information for hosts running TCP/IP and is backward-compatible with MIB-I, though MIB-II supports only an incomplete version of IPv4). The STCP SNMP daemon monitors the STCP-related variables, such as the system variables and variables for the protocol drivers (ARP, ICMP, IP, TCP, and UDP). For complete IPv4 support as well as IPv6 support, see

the *Software Release Bulletin: EMANATE for OpenVOS, Release 17.3 (R612)* for information on the EMANATE for VOS product.

- The SNMP manager (client software) runs on remote network management stations and enables these stations to view and set some of the variables associated with an SNMP daemon in the MIB file. Running the SNMP daemon on your local module enables remote network management stations running a third-party SNMP manager to view the information in the variable MIB file, and to set a few of the variables. (The remote third-party SNMP manager can obtain only information concerning STCP, not any other OpenVOS product.)

The `snmpd` daemon process relies on the UDP protocol and listens for requests on the port specified for `snmp` service in the `services` database file. Port 161 typically defines the `snmp` service. (See the description of the `services` database file earlier in this chapter.)

### How the `snmpconf` File Is Used

The `snmpd` daemon process reads the `snmpconf` database file to determine several system variables that apply to the STCP host. This information is helpful, but is not required by SNMP.

The template `snmpconf` database file resides in the `(master_disk)>system>stcp>templates` directory. Create your own version of the template file and place it in the `(master_disk)>system>stcp` directory.

### How to Edit the `snmpconf` File

The template `snmpconf` file contains entries for SNMP system variables. These variables enable you to supply site-specific information, such as a system contact, system name, system location, and system description.

You can use the `snmpconf` file to change the string that you use to refer to a community.

The file can have comments, preceded by the number sign (`#`), or blank lines. It can contain only printable ASCII characters, and the maximum length is 79 characters.

Table 5-15 describes the system variables in the `snmpconf` file.

**Table 5-15. System Variables in the `snmpconf` File** (Page 1 of 2)

Field	Description
<code>sysContact</code>	Identifies the contact person for this managed host, together with information about how to contact this person (for example, Jane Doe, JDoe@mycorp.com). The maximum length for this text is 255 characters.



**Table 5-15. System Variables in the `snmpconf` File** (Page 2 of 2)

Field	Description
<code>sysName</code>	Specifies the administratively assigned name for the managed host. By convention, this should be the host's fully qualified domain name (for example, <code>m1.mycorp.admin.com</code> ). The maximum length for this text is 255 characters.
<code>sysLocation</code>	Specifies the physical location of the module (for example, <code>Mycorp, Anytown, Ma, Eng, M1-2N</code> ). The maximum length for this text is 255 characters.
<code>sysDescr</code>	Describes the system (for example, <code>Stratus ftServer V 6408, OpenVOS 17.1.0, Streams-based TCP/IP (STCP)</code> ). Use printable ASCII characters to specify the full name and version ID of the system's hardware type, software operating system, and networking software. Note that though this field might not appear in the template <code>snmpconf</code> file, you can specify the field and a value for it.
<code>community_variable</code>	Specifies a value for a community variable. The community variables are as follows:  <pre>community_public community_private community_proxy community_regional community_core</pre>

No other SNMP variables can be set in this file. However, a remote SNMP manager can view and potentially set certain variables that are maintained by the `snmpd` daemon process. See the `snmpd` description in [Chapter 8](#) for more information about `snmpd` and the SNMP variables.

## Sample `snmpconf` File

[Figure 5-15](#) shows the template `snmpconf` database file.

### NOTE

Though the template `snmpconf` file does not include the `sysDescr` field, you can specify the field and a value for it.

```
# This is a sample SNMP configuration file
#
# You should change the file so that it contains relevant information
# about your system.
sysContact = <your_system_administrator and email contact>
sysName = <your_system_name>
sysLocation = <your_system_location>
```

**Figure 5-15. Template *snmpconf* File**

Figure 5-16 shows an example *snmpconf* database file with the fields for *sysDescr* and community strings.

```
# This is a sample SNMP configuration file
#
# You should change the file so that it contains relevant information
# about your system.
sysContact = <your_system_administrator and email contact>
sysName = <your_system_name>
sysLocation = <your_system_location>
sysDescr = <your_system_description>
community_public = <secret_1>
community_private = <secret_2>
community_proxy = <secret_3>
community_regional = <secret_4>
community_core = <secret_5>
```

**Figure 5-16. Example *snmpconf* File with Community Strings**

#### NOTE

---

Daemons and managers can be divided into groups called communities. A remote SNMP manager must set the community to the same string as the *community\_private* string in order to set any writable variables associated with the SNMP daemon.

## The `telnet` service File and TELNET Information

The `telnet` service file enables the `telnetd` daemon process on the module to handle incoming requests (login requests and incoming slave requests) for STCP TELNET services.

The following sections provide additional information about the `telnet` service file.

- [“How the `telnet` service File Is Used” on page 5-53](#)
- [“How to Edit the `telnet` service File” on page 5-55](#)
- [“Sample `telnet` service File” on page 5-56](#)

### NOTE

RSN connections require an entry in the `telnet` service file. For information about this entry, see the manual *OpenVOS System Administration: Configuring a System* (R287).

## How the `telnet` service File Is Used

The `telnetd` daemon process reads the `telnet` service file and creates a TLI transport endpoint for each service listed in the file. Then the `telnetd` daemon process binds a port number to each endpoint according to the port assigned to each service in the file (`master_disk`)>`system`>`stcp`>`services`. Thereafter, the `telnetd` daemon process listens for requests for the specified service at each endpoint. When it receives a request, the `telnetd` daemon process uses an appropriate STCP TELNET window-terminal device to handle the request. Note that the `services` database file **must** list the services that you want the `telnetd` daemon process to invoke at a specified port. (See [“The `services` File and Service Information” on page 5-46](#) for more information.)

Each TELNET service defined in the `telnet` service file must be associated with a number of STCP TELNET devices, which are classified as window-terminal devices. Create a clonable device entry or individual device entries for each STCP TELNET login service and for each STCP TELNET incoming slave service in the `devices.tin` file. (Other STCP TELNET device entries must exist for the TLI access-layer STREAMS driver, the TELNET pipe devices, and any outgoing slave devices. [“Defining STCP TELNET Devices” on page 4-8](#) describes how to create STCP TELNET device entries and how to re-create the `devices.table` file.)

You can optionally use the `-local_ip` field in the `telnet` service file to specify an IPv4 address to which a service binds, in order to restrict the interface on which a service listens. If you do not specify this field, the `telnetd` daemon listens on all interfaces.

The template `telnet` service file, located in the directory `(master_disk)>system>stcp>templates`, defines one basic TELNET login service that accommodates both privileged and nonprivileged users making TELNET connection requests to the module. For many configurations, this single login entry enabling the privileged option may be adequate, and you can simply place a copy of the template file in the `(master_disk)>system>stcp` directory. However, to define additional TELNET services or change an option associated with the basic TELNET login service before you issue the `start_stcp.cm` command macro to start the protocol stack and the `telnetd` daemon process, create your own version of the `telnet` service file and create the appropriate entries.

For example, you can create an entry for a TELNET service using the `linger` option, or for an incoming slave service. The `telnet` service file enables the `keepalive` and `no-delay` options for the default TELNET login service, but consider enabling these options for all your TELNET login and slave services. (The `keepalive` option enables STCP to terminate a dormant connection on the socket associated with the service after periodically transmitting messages to the client. If the client does not respond, STCP terminates the connection. The `no-delay` option tells STCP to send a smaller-sized packet for TELNET as soon as it gets it instead of holding the packet. The `linger` option specifies the number of seconds that STCP maintains a TELNET connection after a close operation in order to provide additional time for pending data to be delivered.)

Do not create entries in this database file for outgoing slave services. The `telnet` service file is only concerned with TELNET services offered locally by the module.

Be sure to place the final version of your `telnet` service file in the `(master_disk)>system>stcp` directory.

#### NOTE

---

To add a TELNET service once the `telnetd` daemon process is running on the module, issue the `telnet_admin` command. (Be sure to add device entries for the service to the `devices.tin` file and re-create the `devices.table` file before you issue `telnet_admin`.) The `telnet_admin` command updates the `telnet` service and `services` files automatically. You can also use this command to delete, modify, or get the status of a TELNET service, as described in [Chapter 9](#).

## How to Edit the `telnet` service File

Enter each entry in the `telnet` service file on a single line. Each entry contains the fields listed in [Table 5-16](#). Separate each field with any number of blank spaces or horizontal tabulation (tab) characters.

**Table 5-16. Fields in the `telnet` service File** (Page 1 of 2)

Field	Description
<code>service_name</code>	Identifies the name of the service, as identified and associated with a port in the <code>services</code> file. The service name can be up to 16 characters long. Required.
<code>device_type</code>	Identifies the device type associated with the service. Specify <code>window_term</code> for the device type. Required.
<code>options</code>	<p>Defines the options for the service in a quoted string. If you do not specify any options, specify a null quoted string to indicate that there are no options for the service. Consider specifying <code>keepalive</code> and <code>nodelay</code> for all of your login and slave connections. (The default login service enables these two options.)</p> <ul style="list-style-type: none"> <li>– Specify <code>keepalive</code> to have STCP test inactive connections by periodically sending probe messages to the client. If STCP does not receive a response from the client, it terminates the connection.</li> <li>– Specify <code>nodelay</code> to have STCP send a smaller-sized packet as soon as it receives it instead of waiting for more data.</li> <li>– Specify <code>linger</code> with a value to specify the number of seconds that STCP maintains a TELNET connection associated with the specified service after a close operation. This provides additional time for pending data to be delivered. The default is 15 seconds if you specify an invalid value for this option. To specify a value for <code>linger</code>, use the format <code>linger=xx</code> (for example, <code>linger=15</code>), without any separating spaces. The value can range from 2 to 32767. When the <code>linger</code> option is disabled, all pending data is discarded when the socket for the connection is closed.</li> </ul>
<code>description</code>	Describes the service using a maximum of 64 characters. Required.
<code>login_slave</code>	Identifies whether the service is a login or slave service. A value of 1 indicates a login service; and a value of 0 indicates a slave service. Required.
<code>privileged</code>	Identifies whether the service (for incoming login connections) handles both privileged and nonprivileged logins, or only nonprivileged. A value of 1 indicates that it handles both privileged and nonprivileged logins and a value of 0 indicates it handles only nonprivileged logins. The value you specify for the service in this field must match the value specified in the device entry or entries associated with the service (the value in the <code>priv_terminal</code> field). Required.

Table 5-16. Fields in the telnet service File (Page 2 of 2)

Field	Description
device_prefix (or device_name)	Identifies either the device prefix or the entire clonable device name associated with the login service or incoming slave service. Each login service or incoming slave service must have either a clonable device name or a device prefix that uniquely identifies any OpenVOS devices that are available for that service. If you are creating a clonable device entry for the service, the entire device name in the device entry must match the name specified in this field. For example, if you assign tli_inslv1m1 for an incoming slave service in the clonable device entry, the name specified in this field must also be tli_inslv1m1. If you decide to use individual device entries instead of using a clonable device entry, then the device name prefix in each device entry for the service (for example, tli_logm1 as the prefix for devices tli_logm1.1, tli_logm1.2, and tli_logm1.3) must match the prefix specified in this field (tli_logm1). Relying on a clonable device prevents you from defining numerous device entries for a given service. The device prefix or clonable device name should not exceed 25 characters. Required.
-local_ip	Specifies the IPv4 address to which the service binds, in order to restrict the interface on which a service listens. If you do not specify this field, the telnetd daemon listens on all interfaces. Optional.

Sample telnet service File

Figure 5-17 shows a telnet service database file with five entries. The first TELNET service is the default login service, which employs the keepalive and no-delay options. This entry appears in the shipped template telnet service database file, but you must add the appropriate module name to the device prefix (or device name). The second entry is for a nonprivileged login service. The other three services are sample entries for incoming slave services, two of which employ the linger option as well as the keepalive and no-delay options. (Note that you must also create entries in the services file for these additional TELNET services.)

```
telnet      window_term "keepalive nodelay"          "Default Telnet login" 1 1 tli_logm1

m1nplog    window_term "keepalive nodelay"          "Nopriv login"        1 0 tli_nplogm1
m1slave1   window_term "keepalive nodelay"          "slave1 service"      0 0 tli_inslv1m1
m1slave2   window_term "keepalive linger=10 nodelay" "slave2 linger"       0 0 tli_inslv2m1
m1slave3   window_term "keepalive linger=15 nodelay" "slave3 linger"       0 0 tli_inslv3m1

my_sv      window_term "keepalive nodelay"          ""                     0 0 my_svc "-local_ip 10.0.2.107"
```

Figure 5-17. Sample telnet service File

## The *xinetd.conf* File and Information for *xinetd*

The *xinetd.conf* file contains default information for the *xinetd* daemon process. Typically, you do not change the *xinetd.conf* file.

To define each service that you want the *xinetd* daemon process to invoke, create a *service* file in the (master\_disk)>system>stcp>*xinetd.d* directory for each service. The name of the file is typically the service name (for example, *tftp6*).

The format of the file contents is as follows:

```
service service
{
    flags          = IPv6
    socket_type    = socket_type
    protocol       = protocol
    wait           = wait_value
    user           = root
    server         = path_name_to_pm
}
```

Table 5-17 describes the fields used in each entry in a *service* file.

**Table 5-17. Fields in a *services* File** (Page 1 of 2)

Field	Description
flags	Specify IPv6 for a service that supports IPv6; otherwise, do not include this field.
socket_type	Identifies the appropriate socket type: <i>stream</i> for a TCP protocol service or <i>dgram</i> for a UDP protocol service.
protocol	Identifies the name of the protocol associated with the service (for example, <i>udp</i> or <i>tcp</i> ).
wait or nowait (mode)	Identifies the mode associated with the service. Specify the value <i>wait</i> if the <i>xinetd</i> daemon process is to suspend listening only on the port until the invoked server has finished executing. Specify the value <i>wait</i> for datagram sockets. Specify the value <i>nowait</i> if the <i>xinetd</i> daemon process is to continue listening on the port after invoking the server (a multithreaded server). Typically, you specify the value <i>nowait</i> for <i>stream</i> sockets.
user	Identifies the user name and group name for the daemon process, enabling you to control the privileges with which the daemon process runs. The <i>user</i> value has the format <i>user_name.group_name</i> . The default user is <i>root.root</i> . However, for security, you may want the user <i>nobody.SysAdmin</i> to run some daemon processes.

**Table 5-17. Fields in a `services` File** (Page 2 of 2)

Field	Description
<code>server</code>	Identifies either the path name of the server program (daemon process) that you want the <code>xinetd</code> daemon process to invoke in order to perform the requested service, or the value <code>internal</code> if <code>xinetd</code> itself provides the service internally.

The following `tftp6` file is an example:

```
service tftp6
{
    flags          = IPv6
    socket_type    = dgram
    protocol       = udp
    wait           = yes
    user           = root
    server         = >system>stcp>command_library>tftpd6.pm
}
```

If you use the `xinetd.conf` file, check the following information in it:

- Confirm that the file contains the following line:  
`includedir /system/stcp/xinetd.d`
- Remove all references to `tcpd` when you convert from the `inetd.conf` file to the `xinetd.conf` file.

See the following web site for information about the `xinetd.d` directory and the `inetd.conf` file, including information about the format of entries in the file:

<http://www-uxsup.csx.cam.ac.uk/pub/doc/redhat/redhat8/rhl-rg-en-8.0/s1-tcpwrap-pers-xinetd.html>



---

## Chapter 6

# Starting, Verifying, and Modifying STCP

This chapter describes how to start, verify, and modify STCP on the module. It contains the following sections.

- “Starting STCP on the Module” on page 6-2
- “Verifying STCP Configuration” on page 6-12
- “Modifying STCP during Normal Operation” on page 6-21
- “Adding, Deleting, or Modifying a TELNET Service” on page 6-23

Table 6-1 summarizes the information in this chapter.

**Table 6-1. Starting, Verifying, and Modifying STCP** (Page 1 of 2)

Desired Operation	What to Do
Setting the STCP command library path	Issue the <code>add_default_library_path</code> and <code>add_library_path</code> commands (from the <code>module_start_up.cm</code> file).
Setting the module's host name	Issue the <code>hostname</code> command (from the <code>module_start_up.cm</code> file).
Starting the STCP protocol stack and many of the daemon processes	Invoke the <code>start_stcp.cm</code> command macro (from the <code>module_start_up.cm</code> file).
Starting the <code>inetd</code> daemon process	Issue the <code>start_inetd</code> command macro as the user <code>root</code> .
Verifying that the STCP protocol stack and related kernel-loadable drivers are loaded	Issue the <code>list_kernel_programs</code> and <code>list_comm_protocols</code> commands.
Verifying basic IP connectivity	Issue the <code>ping</code> command to the loopback interface ( <code>localhost</code> ), then to a host on your local subnet, then to a host that resides across a router.
Verifying TCP connectivity	Issue the <code>telnet</code> or <code>ftp</code> command to start a TELNET or FTP session. The connections should be visible using the <code>netstat -all_sockets</code> command.

**Table 6-1. Starting, Verifying, and Modifying STCP** (Page 2 of 2)

Desired Operation	What to Do
Verifying that the daemon processes are listening at a socket (using <code>netstat</code> ) or generally running (using <code>list_users</code> )	Issue the <code>netstat -all_sockets</code> command and/or the command <code>list_users *.* -process *d -interval -system -admin</code> .
Verifying that the STCP library path is established for the module	Issue the command <code>list_library_paths</code> with the command argument.
Verifying the STCP devices	For TELNET devices, issue the commands <code>list_devices -type streams</code> and <code>list_devices -type window_term</code> . For SDLMUX devices, issue the commands <code>list_devices -type streams</code> and <code>list_devices -type streams_pci</code> .
Adding, verifying, modifying, or deleting a TELNET local service	Issue the <code>telnet_admin</code> command.
Initially configuring or subsequently adding, deleting, or checking the status of a network interface after the stack is up	Issue the <code>ifconfig</code> command.
Stopping the <code>inetd</code> or <code>xinetd</code> daemon process	Issue the <code>stop_inetd.cm</code> or <code>stop_xinetd.cm</code> command macro as a privileged user.

## Starting STCP on the Module

This section discusses the following topics related to starting STCP as the only TCP/IP protocol stack on the module.

- “[Modifying the `module\_start\_up.cm` and `start\_stcp.cm` Files](#)” on page 6-3
- “[Starting STCP during the Current Bootload](#)” on page 6-11

STCP also requires the following STREAMS daemons:

```
Streams_Daemon
Streams_Memory_Daemon
```

All STREAMS drivers on a module require these daemons. You do not start these daemons manually; these daemons start automatically at bootload and whenever a new CPU is added.

## Modifying the `module_start_up.cm` and `start_stcp.cm` Files

To ensure that OpenVOS recognizes several components required by STCP at each bootload, modify the `module_start_up.cm` and `start_stcp.cm` files, as described in the following sections.

- “Establishing the STCP Command Library Paths” on page 6-3
- “Loading SDLMUX” on page 6-3
- “Loading AF UNIX” on page 6-4
- “Establishing the Module’s STCP Host Name” on page 6-4
- “Modifying and Executing the `start_stcp.cm` Command Macro” on page 6-4
- “Starting the `inetd` or `xinetd` Daemon Process” on page 6-9

By convention, many commands exist as comments in OpenVOS command macro files, including the `module_start_up.cm` file, that are shipped on the OpenVOS release media; that is, the commands are preceded by an ampersand (&) and a space. To activate these commands, delete the ampersand and the space preceding each command and specify the arguments required by the command. (In the actual files, the &+ characters at the end of a line indicate a continued command line.)

At each bootload, the `module_start_up.cm` file executes the OpenVOS configuration command `configure_devices` to have OpenVOS recognize the different types of STCP devices as well as all devices associated with the module.

### Establishing the STCP Command Library Paths

To establish the STCP command library, issue the following commands from the `module_start_up.cm` file.

```
add_default_library_path -no_check command >system>stcp>command_library
add_library_path -no_check command >system>stcp>command_library
```

If you are writing an application that uses STCP, define the include and object library paths as well as the command library paths.

For a description of the `add_default_library_path` command, see the manual *OpenVOS System Administration: Administering and Customizing a System* (R281). For a description of the `add_library_path` command, see the *OpenVOS Commands Reference Manual* (R098).

### Loading SDLMUX

If SDLMUX is not already loaded, STREAMS automatically loads SDLMUX when you initialize the first partnered pair of ports of Ethernet PCI adapters. The following

command initializes both the port *device\_name* and its partner (as defined in the `devices.tin` file); the command also loads SDLMUX if it is not already loaded.

```
dlmux_admin #device_name init_sdlmux
```

The `start_stcp.cm` file typically issues the `dlmux_admin` command; thus, you need to customize and activate the `dlmux_admin` command line as it appears in the `start_stcp.cm` file that you receive on the OpenVOS release media.

## Loading AF UNIX

To configure the `af_unix` driver on OpenVOS, the `module_start_up.cm` file must include the following command:

```
configure_comm_protocol af_unix
```

If the `module_start_up.cm` file on your system does not already contain this command line, you can copy it from the `sample_module_start_up.release_no` file in the `(master_disk)>system>configuration` directory. Add the `af_unix` line just after the command lines that start the Overseer.

## Establishing the Module's STCP Host Name

To permanently define the host name for the appropriate module (typically, the local host), issue the `hostname` command from the `module_start_up.cm` file. The following sample `hostname` command defines the STCP host name for local host #m1.

```
hostname m1
```

Once you issue the command, the command creates a file called `host` in the `(master_disk)>system>stcp` directory and places the STCP host name in it. Make sure that this file is never deleted.

See [Chapter 9](#) for more information about the `hostname` command.

## Modifying and Executing the `start_stcp.cm` Command Macro

To start STCP on a Stratus module automatically at each bootload, execute the `start_stcp.cm` command macro from the `module_start_up.cm` file. The `start_stcp.cm` command macro loads SDLMUX, starts the STCP protocol stack, configures network interfaces and routes, and starts several of the daemon processes.

A template `start_stcp.cm` command macro resides in the directory `(master_disk)>system>stcp>templates`. Copy the template file and modify the copy for your own configuration.

### NOTE

You must modify the `start_stcp.cm` file before executing the pre-boot phase of an OpenVOS software

installation and before rebooting the module. **Otherwise, your network connection may fail to restart properly.**

Review the configuration checklists in [Chapter 2](#) and edit your version of the `start_stcp.cm` file to create the STCP configuration that is appropriate for your system. (Do not place any `dlnux_admin` commands or STCP `ifconfig` or `route` commands in the `module_start_up.cm` file. These commands should appear only in the `start_stcp.cm` command macro.)

Modify the template `start_stcp.cm` command macro as follows:

- Stratus recommends that you start with the template provided in OpenVOS Release 18.x or later and add your configuration specific commands. The template contains new code to check if the STCP stack has already been loaded. However, if you are working with your own copy of the `start_stcp.cm` file **from a release prior to 18.0.0**, it contains old STCP stack checking code (shown below). You must ensure that the following lines are deleted or commented out (by inserting an ampersand and a space at the beginning of each line) and that the new stack checking code is included. Otherwise, **your network connections may fail to restart properly.** (A plus (+) sign indicates a wrapped line.)

```
& First check if STCP is already loaded, if it is, then exit
& !attach_default_output (master_disk)>system>stcp>query
& &if (command_status) ^= 0
& &then &goto other_errors
& !list_comm_protocols
& !detach_default_output
& !display -match '(ip.cp.pm)' (master_disk)>system>stcp>query
+ -output_path (master_disk)>system>stcp>temp
& &if (command_status) ^= 0
& &then &goto other_errors
& &if (file_info (master_disk)>system>stcp>temp blocks_used) = 0
& &then &goto Already_Loaded
```

- Check that the `start_stcp.cm` command macro executes STCP stack options (for example, the commands `ifconfig` and `route`) immediately after executing the `start_stcp_stack` command. In the template `start_stcp.cm` command macro, the command `IP_forwarding off` appears at the beginning of the section of commands that execute STCP stack options.
- Edit the command lines `dlnux_admin device_name init_sdlmux` to load SDLMUX (and its lower-level device drivers) and to configure the SDLMUX-associated devices if you are using SDLMUX-associated devices.
- Edit the `ifconfig` command lines, creating a command line to define each logical interface required by your system.

Each `ifconfig` command associates an IPv4 or IPv6 address to an SDLMUX interface. You can associate multiple addresses with an interface using the `-alias` argument. Interfaces are enabled for IPv4 and IPv6 traffic by default. If you do not want to enable an interface for IPv6, use the `-no_ipv6` argument. Interfaces enabled for IPv6 are assigned one or more addresses automatically, where the first address is a link-local address. In addition, a process called *stateless auto-configuration* might also define one or more global IPv6 addresses. Consequently, you need to edit `ifconfig` command lines for IPv6 addresses only if you need a specific value. Stateless auto-configuration is controlled by the router(s) on the interface's subnet. You can disable stateless auto-configuration on an interface using the `-no_accept_router_adv` argument of the `ifconfig` command.

#### NOTE

---

When you configure an interface using the STCP `ifconfig` command, STCP automatically creates a route for the **local** network/subnetwork, based on the network address and subnet mask of a logical interface. These local routes are displayed by the command `route print`.

- If you are adding IPv6 protocol support to an existing `start_stcp.cm` command macro, copy the following lines from the new template `start_stcp.cm` command macro to the existing macro, to appear immediately before the `route` commands:

```
&
&    Wait for DAD to complete in the last interface we
ifconfig'd
&
!(master_disk)>system>command_library>sleep -seconds (calc 1
+ (system_info 1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.1.4.109))
```

You do not need to modify the `sleep` command line.

- Edit the appropriate `route` commands to define routes for modules that are **not** on the same subnets. For an IPv4 router, you typically define a default route for use by peers that are not on the local subnet.

#### NOTE

---

In your version of the `start_stcp.cm` file, the order of commands must be identical to the order of commands in the `start_stcp.cm` template file (in the directory `(master_disk)>system>stcp>templates`). This

order is:

- the `start_stcp_stack` command
- set options (for example, `IP_forwarding`)
- the `dlnux_admin` command
- the `ifconfig` command
- the `sleep` command
- the `route` command
- the `rtssold` command
- command lines for other daemon processes

You do not need to define a default route for IPv6: with IPv6 protocol support, routers are found automatically. You can modify this process, though, with the `ifconfig` command and in the configuration of the routers.

- Edit command lines associated with the `rtssold` daemon process to start the router solicitation daemon for IPv6 protocol support. The `rtssold` daemon process should be passed a list of all the interfaces using IPv6 that are not configured in a router. The command lines to start the `rtssold` daemon process must appear after all of the interfaces have been configured, but before starting any other daemons. For additional information about enabling IPv6 support, see “[Checklist for Enabling IPv6 Support](#)” on page 2-9.

The `start_stcp.cm` command macro also contains the following line, which invokes the `mddmon_start_up.cm` file.

```
(master_disk)>system>mddmon_start_up.cm
```

The `mddmon_start_up.cm` command macro contains command lines that start the RSN (`start_rsn`), as well as maintenance and diagnostic daemons for the network I/O enclosure (`diag_nio_monitor.pm`) and the UPS (`diag_ups_monitor.pm`). The `mddmon_start_up.cm` file then returns control to `start_stcp.cm`.

Place your version of the `start_stcp.cm` command macro in the `(master_disk)>system>stcp` directory. The `start_stcp.cm` command macro **must** reside in this directory.

### Starting the Stack and the Daemon Processes

By default, the `start_stcp.cm` command macro performs a series of operations, as follows:

- `start_stcp.cm` creates the `(master_disk)>system>stcp>logs` directory to hold all of the log files generated by the `start_stcp_stack` command and the daemon processes.
- `start_stcp.cm` issues the `dlnux_admin #device_name init_sdlnux` commands for SDLMUX-associated network interfaces.

- `start_stcp.cm` issues the internal `start_stcp_stack` command to bring up the STCP stack.

If an error occurs while the stack is being built, the `start_stcp.cm` command macro cancels building the stack and displays an error message stating that the stack could not be brought up. It also displays a message that instructs you to check the contents of the `start_stcp_stack.out` file, which is created in the `(master_disk)>system>stcp>logs` directory, for information about the errors that occurred.

- `start_stcp.cm` issues `ifconfig` commands that you specify to add and start each network interface (each logical interface, from the perspective of the STCP stack).
- `start_stcp.cm` issues `route` commands that you specify to add STCP routes.
- `start_stcp.cm` issues a `start_process` command to start the `telnetd` daemon process, which enables the module to handle incoming requests for STCP TELNET services.
- `start_stcp.cm` issues a `start_process` command to start the `ftpd` daemon process, which enables the module to handle incoming FTP requests.
- `start_stcp.cm` issues a `start_process` command to start the `snmpd` daemon process, which enables network-management support.
- `start_stcp.cm` issues a `start_process` command to start the `ospfd` daemon process, which allows the module to learn routes from OSPF.
- `start_stcp.cm` issues a set of commands to ensure that each daemon process sends status and error messages to the corresponding log file. When the stack comes up, messages go to the appropriate log file (for example, `telnetd.out`); if the module is rebooted, a new log file is generated and the old log file is renamed to include a date and time stamp (for example, `telnetd.01-02-14.22:12:26.out`) and archived. (You can view even the most recent log files while the daemon processes are running, which enables you to get troubleshooting information immediately after seeing a problem.)

#### NOTE

You must use the `start_stcp.cm` command macro to issue the commands to start the protocol stack, most of the daemon processes, and the network interfaces during the current bootload. The `start_stcp_stack` command called by `start_stcp.cm` is an internal command that should only be issued by Stratus support personnel, usually for debugging purposes.



- The `start_stcp.cm` file starts the `inetd` daemon process, by default. If, instead, you want to start the `xinetd` daemon process, see “[Starting the `inetd` or `xinetd` Daemon Process](#)” on page 6-9.
- `start_stcp.cm` issues a `start_process` command to start the `rtssold` daemon process, which sends ICMPv6 Router Solicitation messages on specified interfaces.

For a description of the `ifconfig` command, see [Chapter 9](#); for a description of the daemon processes, see [Chapter 8](#).

### Specifying Routes

In the `start_stcp.cm` command macro, you can specify the routes for your configuration using the `route` command. The template `start_stcp.cm` command macro shows a sample `route` command (commented out) with a sample IP address.

Specify your own `route` command lines; do not simply uncomment the sample `route` command, since it is intended only as an example of one form of the command. For example, to specify network routes, specify the correct subnet mask (such as the subnet mask for a variable-length subnet (VLSN) configuration).

Issuing a `route` command from the `start_stcp.cm` command macro sets the route at each bootload and builds a routing table for your configuration (which you can later display using the `route` command with `print` for the `command` argument or using the `netstat` command with the `-routing` argument).

The format of the command depends on your specific configuration. You may need to add a route to a specific destination through a router/gateway, or a route to any number of unspecified destinations through a default router/gateway. For example, to add a route that enables module #m1 to communicate with the hosts on the networks and subnets available through the local router/gateway 172.16.2.20, issue the following `route` command, which establishes a default router/gateway route in the routing table. This default route frees you from having to know the other routes that 172.16.2.20 uses to reach a destination.

```
route add * 172.16.2.20 -default_gateway
```

For more information about the `route` command, see the command description in [Chapter 9](#). For more information about STCP routes, see “[STCP Routes](#)” on page A-18.

### Starting the `inetd` or `xinetd` Daemon Process

The `start_stcp.cm` file starts the `inetd` daemon process by default, by executing the `start_inetd.cm` file. To enable the `start_stcp.cm` file to start the `xinetd` daemon process, instead, specify the value `yes` for the `-xinetd` argument when invoking the `start_stcp.cm` command macro. The value `yes` for the `-xinetd` argument enables the `start_stcp.cm` file to execute the `start_xinetd.cm` file.

To enable the `start_stcp.cm` file to start the `xinetd` daemon process from the `module_start_up.cm` file, add the `-xinetd` argument to the `start_stcp.cm` command line in the `module_start_up.cm` file.

You can also manually start either daemon process by issuing the start command macro for either daemon:

- The `start_inetd.cm` file starts the `inetd` daemon process—[Figure 6-1](#) shows the template `start_inetd.cm` file.

```
& *****INETD*****
&begin_parameters
&end_parameters privileged

start_process -output_path (master_disk)>system>stcp>logs>inetd.out
+'(master_disk)>system>stcp>command_library>inetd' -privileged -root
+-process_name stcp_inetd
!display_line STCP_INETD .....
```

**Figure 6-1. Template `start_inetd.cm` File**

- The `start_xinetd.cm` file starts the `xinetd` daemon process—[Figure 6-2](#) shows the template `start_xinetd.cm` file.

```
& *****XINETD*****
&begin_parameters
&end_parameters privileged

&mode no_abort

!start_process -output_path (master_disk)>system>stcp>logs>xinetd.out
+'!(master_disk)>system>stcp>command_library>xinetd -f
+(master_disk)>system>stcp>templates>xinetd.conf' -privileged -root
+-process_name xinetd
!display_line XINETD .....
```

**Figure 6-2. Template `start_xinetd.cm` File**

Both of these start command macro files reside in the directory `(master_disk)>system>stcp`.

## Starting STCP during the Current Bootload

After you permanently configure the module for STCP only, you can start the different portions of STCP.

To start STCP on the module, do one of the following:

- Reboot the module immediately to have OpenVOS execute the `module_start_up.cm` file, which you have modified to permanently establish STCP.
- Issue the appropriate OpenVOS commands from command level to start STCP during the current bootload.

If you want to use the `inetd` daemon process, you must also issue the `start_inetd.cm` command macro as the user `root` to start the process.

The following list summarizes how to start STCP during the current bootload. (The software should already be installed from an OpenVOS release media.)

1. Make sure that the `devices.tin` file has entries for the SDLMUX and STCP devices, and that you generate a new `devices.table` file by issuing the `create_table` command. Then, issue the `broadcast_file` command to copy the new file to the `(master_disk)>system` directory and issue the `configure_devices` command to activate the new table.
2. Make sure that the database files contain the appropriate information about your configuration. The template files reside in the `(master_disk)>system>stcp>templates` directory. Use the templates to create your own files.
3. Issue the commands that add the library paths of the STCP command library.
4. Issue the `hostname` command to define the host name for the module.
5. Make sure that the `start_stcp.cm` command macro has the correct information (the `dlmux_admin #device_name init_sdldmux` command lines **as well as** `ifconfig` commands for each logical network interface, which must appear after the command lines for the protocol stack).
6. Issue the `start_stcp.cm` command macro to start STCP on the module. (If you did not place `route` commands in the command macro, issue `route` commands to define any routes.)
7. Issue the `start_inetd.cm` command macro as the user `root` to start the `inetd` daemon process.

## Verifying STCP Configuration

After you issue the appropriate configuration commands or reboot the module, confirm that STCP is operational. You can also use standard OpenVOS commands to verify that the individual STCP components are initialized and that the associated communications drivers and devices are properly configured. The following sections describe how to verify the different aspects of your STCP configuration.

- [“Verifying the STREAMS Daemons” on page 6-12](#)
- [“Verifying the STCP Command Library” on page 6-13](#)
- [“Verifying the Host Name for the Module” on page 6-13](#)
- [“Verifying the Components in the Kernel-Loadable Library” on page 6-13](#)
- [“Verifying the Communications Protocols” on page 6-14](#)
- [“Verifying the Drivers” on page 6-15](#)
- [“Verifying the Networking Daemon Processes” on page 6-16](#)
- [“Verifying the STCP Devices” on page 6-16](#)
- [“Verifying Network Interfaces” on page 6-18](#)
- [“Verifying Entries in the Routing Table” on page 6-19](#)
- [“Verifying Basic IP Connectivity” on page 6-19](#)
- [“Verifying TCP Connectivity” on page 6-20](#)
- [“STCP Requests of the analyze\\_system Subsystem” on page 6-21](#)

See [Chapter 10](#) for information about how to resolve problems that you may encounter with STCP and associated applications.

## Verifying the STREAMS Daemons

To verify that the STREAMS daemons (`Streams_Daemon` and `Streams_Memory_Daemon`) are running, issue the following command:

```
list_users -kernel -process streams*
```

When these daemons are running, the command output includes the following lines:

```
Overseer.System (Streams_Daemon)
Overseer.System (Streams_Daemon)
Overseer.System (Streams_Memory_Daemon)
```

## Verifying the STCP Command Library

To confirm that the STCP command library path is included in the default command library paths for the module, issue the OpenVOS commands `list_library_paths` and/or `list_default_library_paths`. The output of both commands should include the following library path.

```
%s1#m1>system>stcp>command_library
```

See the *OpenVOS Commands Reference Manual* (R098) for a description of the `list_library_paths` command. See the manual *OpenVOS System Administration: Administering and Customizing a System* (R281) for a description of the `list_default_library_paths` command.

## Verifying the Host Name for the Module

To confirm that the host name for the module has been defined, issue the STCP `hostname` command (without arguments). For example, the command displays the following information for module #m1.

```
s1m1
```

[Chapter 9](#) describes the `hostname` command and provides sample output.

## Verifying the Components in the Kernel-Loadable Library

To verify that all of the STCP and related drivers have been loaded from the `(master_disk)>system>kernel_loadable_library` directory, issue the OpenVOS `list_kernel_programs` command.

The following sample `list_kernel_programs` command indicates which drivers have been loaded on sample module #m1. This configuration uses SDLMUX.

- the STREAMS drivers needed to build this module's STCP stack (`stcp.cp.pm`, `udp.cp.pm`, `ip.cp.pm`, `adp.cp.pm`, and `arp.cp.pm`)
- the OpenVOS STREAMS driver (`timod.cp.pm`) required for TLI configuration (`ftp` and `telnetd`)
- the SOSL Net driver (`sosl_net_driver`) required for Open StrataLINK
- the STCP TELNET-related drivers (`tnmod.cp.pm` and `tpipe.cp.pm`), which are STREAMS drivers

- the related SDLMUX drivers (sdlmux.cp.pm as well as igb.cp.pm or ixgbe.cp.pm) in configurations using SDLMUX

#### **list\_kernel\_programs**

Loaded Kernel Programs:

```
%sw#m1>system>kernel_loadable_library>timod.cp.pm
%sw#m1>system>kernel_loadable_library>tnmod.cp.pm
%sw#m1>system>kernel_loadable_library>af_unix.cp.pm
%sw#m1>system>kernel_loadable_library>ip.cp.pm
%sw#m1>system>kernel_loadable_library>sdlmux.cp.pm
%sw#m1>system>kernel_loadable_library>adp.cp.pm
%sw#m1>system>kernel_loadable_library>tpipe.cp.pm
%sw#m1>system>kernel_loadable_library>sosl_net_driver.cp.pm
%sw#m1>system>kernel_loadable_library>streams_pipe.cp.pm
%sw#m1>system>kernel_loadable_library>vterm.pm
```

The OpenVOS `list_kernel_programs` command is described in the manual *OpenVOS System Administration: Administering and Customizing a System* (R281).

## **Verifying the Communications Protocols**

To verify that the STCP drivers and other communications drivers (protocols) are configured on the module, issue the OpenVOS `list_comm_protocols` command. When you issue the `list_comm_protocols` command, the command lists the communications protocols (drivers) loaded on the module.

The following example, which is from a module running OpenVOS Release 18.0.0, is a list of all communications drivers loaded on the module, including the individual STCP drivers (such as `stcp`, `ip`, `loop`, and `udp`), along with the TELNET components (`timod`, `tnmod`, and `tpipe`) loaded by STREAMS to service TELNET requests. The example also includes the `af_unix` driver.

Communication Protocol Drivers:

```
3270_host..... loaded
3270_remote (3270_remote.pm) .....
SDLC_host..... loaded
adp..... loaded
af_unix..... loaded
arp..... loaded
bisync..... loaded
cpc_lapb (cpc_lapb.cp.pm) .....
eft_bsc..... loaded
generic_comm..... loaded
```

(Continued on next page)

```

hasp..... loaded
igb..... loaded
ip..... loaded
ipv6..... loaded
ixgbe..... loaded
loop..... loaded
mpx_gcomm..... loaded
pci_gcomm (pci_gcomm.pm) .....
poll_select..... loaded
printer..... loaded
pt_tioc_mod..... loaded
random (random.cp.pm) .....
sdlmux..... loaded
server (server_device.pm) .....
sosl_net_driver..... loaded
stcp..... loaded
streams..... loaded
streams_pipe..... loaded
term_dil..... loaded
terminal..... loaded
timod..... loaded
tnmod (tnmod.cp.pm) ..... loaded
tpipe..... loaded
udp..... loaded
urandom (urandom.cp.pm) .....
usb (usb.cp.pm) .....
visa..... loaded
vterm..... loaded
window_term..... loaded
x25_lap..... loaded

```

All STREAMS drivers are loaded (and unloaded) by the OpenVOS STREAMS software on the module. For a description of the `list_comm_protocols` command, see the manual *OpenVOS System Administration: Configuring a System* (R287).

## Verifying the Drivers

To confirm that the protocol drivers are loaded and able to handle connections, use the `analyze_system` subsystem, as the following example shows.

```

analyze_system
as: match ip.cp.pm; dump_eit -summary
as: C0E0C480 %eng#m1>system>kernel_loadable_library>ip.cp.pm

```

## Verifying the Networking Daemon Processes

To confirm that the daemon processes are listening at the appropriate ports, issue the `netstat` command with the `-all_sockets` argument. The following example shows sample `netstat` output.

```
netstat -all_sockets
```

```
Active connections (including servers)
Proto Recv-Q Send-Q Local Address      Foreign Address    (state)
tcp    0      0 *:ftp              *:*                LISTEN
tcp    0      0 *:echo             *:*                LISTEN
tcp    0      0 *:discard          *:*                LISTEN
tcp    0      0 *:daytime          *:*                LISTEN
tcp    0      0 *:chargen          *:*                LISTEN
tcp    0      0 *:time             *:*                LISTEN
tcp    0      0 *:telnet           *:*                LISTEN
tcp    0      84 172.16.99.38:telnet 172.16.2.12:1635  ESTABLISHED
udp    0      0 *:snmp             *:*                *
udp    0      0 *:echo             *:*                *
udp    0      0 *:discard          *:*                *
udp    0      0 *:daytime          *:*                *
udp    0      0 *:chargen          *:*                *
udp    0      0 *:time             *:*                *
udp    0      0 *:tftp             *:*                *
udp    0      0 *:bootps           *:*                *
```

You can also use the OpenVOS `list_users` command to verify that the daemon processes are running, as shown in the following example.

```
list_users *.* -process *d -admin -system
USERS at 10:01:03      CPU  DCPU    PF  DPF M PS F P PROGRAM
Overseer (ftpd)        0.0          65    0 WS + 7 ftpd.pm
Overseer (ospfd)       1:16.2      171    0 WS + 7 ospfd.pm
Overseer (snmpd)       0.0          61    0 WS + 7 snmpd.pm
Overseer (telnetd)     0.2          82    0 WS + 7 telnetd.pm
root (stcp_inetd)      0.5          89    0 WS + 7 inetd.pm
```

See the *OpenVOS Commands Reference Manual* (R098) for a description of the `list_users` command.

## Verifying the STCP Devices

To verify that the different types of OpenVOS devices associated with STCP are configured on the module, issue the OpenVOS `list_devices` command, once for the STREAMS devices and a second time for the TELNET devices.



The following sections describe the OpenVOS devices that are associated with STCP.

- [“Verifying the STREAMS Devices” on page 6-17](#)
- [“Verifying the TELNET \(Window-Terminal\) Devices” on page 6-18](#)

### Verifying the STREAMS Devices

To confirm configuration of the STREAMS devices associated with STCP (for example, the STCP protocol drivers and the network interfaces running STCP), issue the following OpenVOS `list_devices` command:

```
list_devices -type streams
```

The following example shows the output of the preceding command. Note that some STREAMS clone devices for the protocol modules, such as IP, UDP, and STCP, are listed.

```
%s1#enetA.m1.10-6.0
%s1#enetA.m1.11-6.0
%s1#ip.m1
%s1#stcp.m1
%s1#udp.m1
%s1#loop.m1
%s1#tpipe_telnetd.m1
%s1#tpipe_admin.m1
%s1#udp.m1_1
%s1#ip.m1_3
%s1#stcp.m1_1
%s1#stcp.m1_2
%s1#udp.m1_2
%s1#stcp.m1_3
%s1#udp.m1_3
%s1#stcp.m1_4
%s1#udp.m1_4
```

To list a specific type of network interface, issue the `list_devices` command for that type of device. For example, `list_devices -type streams_pci` lists the ports of Ethernet PCI adapters, which use SDLMUX.

For information about a network interface, issue the `ifconfig` command, or the `netstat` command with the `-interface` argument and the device name of the interface. To confirm the status of partnered network interfaces, issue the `dlnux_admin` command.

## Verifying the TELNET (Window-Terminal) Devices

The STCP TELNET login and slave devices are classified as window terminal devices. To display a list of the configured window terminal devices, issue the following OpenVOS `list_devices` command:

```
list_devices -type window_term
```

The following example shows the output of the preceding command. It lists the window terminal devices configured on the module `%s1#m1`. In this example, the only window terminal devices are the STCP TELNET devices.

```
%s1#tli_logm1_1
%s1#tli_logm1_2
%s1#tli_logm1_3
%s1#tli_logm1_4
%s1#tli_logm1_5
%s1#tli_inslv1m1_1
%s1#tli_inslv1m1_2
%s1#tli_outslv1m1
```

For a description of the `list_devices` command, see the *OpenVOS Commands Reference Manual* (R098).

## Verifying Network Interfaces

To verify the state of a network interface, issue the `ifconfig` command. The following example shows how you can check the status of the interface by issuing the `ifconfig` command with only the device name of the interface.

```
ifconfig #enet.m1.4.11
```

```
#enet.m1.4.11: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST, KEEPALIVE>
172.16.2.14 netmask 0xffffffff broadcast 172.16.2.25
```

The output indicates the following:

- UP indicates that the network interface is administratively configured to be operational.
- BROADCAST indicates that a broadcast address is set.
- RUNNING indicates the interface's operational line state (that is, the interface is actively handling the communications I/O).
- NOFORWARDBROADCAST indicates that the forward broadcast flag is disabled.
- KEEPALIVE indicates that the keepalive option is enabled.

To display a list of the network interfaces configured for STCP, issue the `ifconfig` command with no arguments or with the `-all` argument.

## Verifying Entries in the Routing Table

To check the contents of the routing table, issue either the `netstat` command with the `-routing` argument or the `route` command with the `print` argument. The output of either command displays any STCP routes that have been set:

- by `ifconfig`
- statically, with the `route` command
- dynamically, by OSPF
- statically, with the `omon` subcommand, `add_static_route`
- using ICMP redirect messages

The following example shows a sample `route` command that displays the routing table. In this example, the routing table reflects receipt of an ICMP redirect that redirects the network route `172.16.13.0` to another router/gateway (`172.16.12.20`).

```
route print
```

Network Address	Gateway Address	Subnet Mask	Redirect	Life
172.16.13.0		255.255.255.0	172.16.12.20	5 mins

See [Chapter 9](#) for descriptions of the `netstat` and `route` commands with sample output.

## Verifying Basic IP Connectivity

To verify that STCP is operational on your module, issue a series of `ping` commands, as follows:

1. To verify STCP communication to the loopback interface, which has the host name `localhost` in the `hosts` file and the IP address `127.0.0.1`, issue the following command:

```
ping localhost
```

If the preceding command is successful, STCP is working on your local host; if it fails, STCP is not operational, and you should perform the troubleshooting procedures defined in [Chapter 10](#).

2. To verify that you can reach a host on your local subnet, issue a command similar to the following:

```
ping 172.16.2.47
```

If the preceding command is successful, STCP is operational across a network interface to your local subnet. If the command fails (but you could `ping` your local host), you cannot reach the local subnet. See [Chapter 10](#) for instructions.

3. To verify that you can use host names (that is, to verify that the `hosts` file contains the appropriate definitions or that the DNS server on the network is working), issue the `ping` command to a remote host on the LAN using the host name. For example:

```
ping m11
```

If the preceding command returns the following message but you can use the IP address, there is a name resolution problem.

```
ping m11
m11: unknown host
```

4. To verify that you can reach a remote host that is not directly on your local subnet, issue a `ping` command to a remote host that resides on another subnet but is accessible through a router/gateway. For example:

```
ping 172.16.3.56
```

If the preceding command is successful, the routing is working correctly. If not, consult [Chapter 10](#) for troubleshooting information.

## Verifying TCP Connectivity

To verify that you can establish TCP connectivity, issue a command such as `telnet` (or `ftp`) to reach another host on your local subnet. (You must ensure that the remote host is running the appropriate TELNET and/or FTP daemon process.) For example:

```
telnet 172.16.2.35
```

If the preceding command is successful, it establishes a TCP TELNET connection. You can verify the connection using the `netstat -all_sockets` command, which lists all TELNET and FTP connections.

## STCP Requests of the `analyze_system` Subsystem

The `analyze_system` subsystem includes the following STCP-related requests:

- `set_stcp_param` and `list_stcp_params`—These requests allow you to list and set various STCP parameters.
- `stcp_meters`—This request displays various STCP meters in order to allow you to analyze STCP performance and activity.

For information on using these requests to set and view security-related parameters, see [Chapter 7, “STCP and Security,”](#) which includes sample output of the `list_stcp_params` request. For more information about these and other requests of the `analyze_system` subsystem, see the *OpenVOS System Analysis Manual* (R073).

## Modifying STCP during Normal Operation

Aside from stopping an individual daemon process (with the `stop_process` command), you can modify certain components of STCP during normal operation. The following sections describe how to modify STCP during normal operation.

- [“Adding, Deleting, or Changing the State of a Network Interface” on page 6-21](#)
- [“Adding, Deleting, or Modifying a TELNET Service” on page 6-22](#)

### Adding, Deleting, or Changing the State of a Network Interface

Use the `ifconfig` command during normal operation to:

- add a network interface to STCP
- delete a network interface from STCP
- change the data-transfer state of a network interface for STCP (that is, change the data-transfer state of a network interface to `up` or `down`)
- verify a specific network interface or all network interfaces configured for STCP
- add an IP address to a network interface

Do not issue this command to modify the flags of an existing (already configured and operational) network interface. To change the setting of a flag for a network interface (for example, the subnet mask), issue the `ifconfig` command to delete the interface and then issue the `ifconfig` command to re-add the interface, specifying the new flag.

Before you issue the command to add or delete a network interface, make sure that a device entry exists for the network interface in the activated `devices.table` file. To add the interface, create a device entry in the `devices.tin` file, re-create the

devices.table file and place it in the (master\_disk)>system directory, and then issue the OpenVOS `configure_devices` command to put the new devices.table file into effect.

To add a network interface, issue the `ifconfig` command with the `-add` argument, specifying the device name and IP address associated with the network interface, as well as the appropriate subnet mask. You can also specify various options (for example, `-forwb`, which enables the module to forward broadcasts). Note that when you add a network interface, the command marks the state of the network interface `up`.

```
ifconfig #enet.m1.6 172.16.3.16 -netmask 0xffffffff00 -add
```

For more information about adding SDLMUX devices, see [Chapter 3](#).

To delete a network interface, issue the `ifconfig` command with the `-delete` argument, specifying the device name that represents the network interface. By default, the command prompts you to make sure that you want to proceed with the deletion. Note that when you delete a network interface, the command automatically marks the state of the network interface `down`.

If you are adding the interface permanently, be sure to update the `start_stcp.cm` command macro with a corresponding `ifconfig` command. This will enable STCP to capture the addition each time the stack is started. If you are permanently deleting a network interface, delete the entry for the interface from the `start_stcp.cm` command macro. If you do not modify the `start_stcp.cm` command macro to reflect the addition or deletion, the addition or deletion will be temporary; that is, it will be in effect only until the module is rebooted.

To add an additional IP address to a network interface, issue the `ifconfig` command with the `-add` and `-alias` arguments, and also specify the device name, IP address, and netmask associated with the network interface, as shown in the following example.

```
ifconfig #enet.m1.4.11 172.16.2.20 -netmask 0xffffffff00 -add  
-alias
```

For more information about the `ifconfig` command, see [Chapter 9](#).

## Adding, Deleting, or Modifying a TELNET Service

Use the `telnet_admin` command to perform the following operations while STCP is running on the module.

- add a local TELNET service
- delete a local TELNET service
- change the options associated with a local TELNET service
- verify your local TELNET services

When you issue the `telnet_admin` command to add, delete, or change a local TELNET service, the command automatically updates the `telnetSERVICE` and `SERVICES` database files. Therefore, the addition, deletion, or change is considered permanent (unless you change the service again).

Although there is only one `telnetd` daemon process, each service must be associated with a unique port number. (The default TELNET service for logins uses port 23.) Therefore, review your existing `SERVICES` database file and select an unused port number before you add a TELNET service.

Before you issue the command to add a TELNET service, make sure that one or more device entries exist to accommodate the service in the activated `devices.table` file. Create the device entries that use the login or incoming slave service in the `devices.tin` file, re-create the `devices.table` file and place it in the `(master_disk)>system` directory. Then issue the OpenVOS `configure_devices` command to put the new `devices.table` file into effect.

Remember that device entries for an incoming slave service must contain a clonable device name or a common device prefix. Specify this clonable device name or prefix with the `telnet_admin` command, as well as a service name and port number. In addition, for login or slave devices, the values in the `login_slave` and `priv_terminal` fields of the device entries must match the values you supply for the login or slave service with the `telnet_admin` command. (Outgoing slave devices also require entries in the `devices.tin` file, but these devices do not have entries in the `telnetSERVICE` database file since their service is not available on the local module. The module actually providing that service must contain an entry in its own database file.)

The following example adds an incoming slave TELNET service with the `telnet_admin` command.

```
telnet_admin -service m1slave1 -device_prefix tli_inslv1m1
             -description 'slave service 1' -no_login -no_privileged
             -services_port 1101
```

See [Chapter 9](#) for a description of the `telnet_admin` command. See [Chapter 5](#) for a description of the `telnetSERVICE` and `SERVICES` database files.





---

## Chapter 7

# STCP and Security

STCP provides security-related parameters for dealing with attacks from malicious users at external sites. You view and set these parameters using the `list_stcp_params` and `set_stcp_param` requests of the `analyze_system` command. The security-related parameters that you can set are as follows:

```
log_syn_attacks
detect_syn_attack
prevent_port_scan
```

The security-related parameter that is informational only (you cannot set it) is `syns_discarded`.

Use the `list_stcp_params` request to list the existing values of the parameters. Use the `set_stcp_param` request to change the values of the settable parameters.

This chapter, which contains the following sections, discusses how to use the security-related parameters.

- [“LAND Attack” on page 7-2](#)
- [“Denial-of-Service Attack” on page 7-2](#)
- [“Logging Discarded Packets” on page 7-3](#)
- [“Port Scanning” on page 7-4](#)
- [“Excerpt from `list\_stcp\_params` Output” on page 7-4](#)

For complete information on the `list_stcp_params` and `set_stcp_param` requests, see the *OpenVOS System Analysis Manual* (R073).

STCP also provides other types of security:

- You can control access to the STCP driver and other STCP devices (see [“Controlling Access to STCP Devices” on page 4-30](#)).
- The daemons `telnetd`, `ftpd`, and `tcpd` for `inetd` can authenticate client access using the `hosts.allow` and `hosts.deny` files (see [“The `hosts.allow` and `hosts.deny` Files and TCP Wrappers” on page 5-10](#)).

- In the `telnet.service` file, you can specify an IP address to which a service binds, in order to restrict the interface on which a service listens. For information on the `telnet.service` file, see [“The telnet.service File and TELNET Information” on page 5-53](#).

## LAND Attack

One type of breach of computer security is commonly referred to as a LAND attack. In a *LAND attack*, a malicious client sends a TCP SYN packet in which the source IP address and TCP port number are identical to the destination address and port number. In early TCP implementations, these packets could cause the host to loop, sending packets to itself.

Legitimate TCP implementations do not create these types of packets. A malicious user creates a LAND packet typically using the raw socket interface. STCP is not susceptible to LAND attacks and does not attempt to respond to such packets. By default, STCP logs a message to the system error log file (`syserr_log.date`) when it receives such a packet (see [“Logging Discarded Packets” on page 7-3](#)).

## Denial-of-Service Attack

One type of attack on a host is a denial-of-service (DOS) attack. In a *DOS attack*, a malicious client (or group of clients) sends a large number of connect requests (that is, SYN packets) to a host, with the intention of overwhelming the host’s capacity, thereby reducing the host’s ability to respond to legitimate requests. STCP is optimized to quickly dispose of such requests, so that the impact on performance is not severe enough to make the system non-responsive to other activities or to legitimate connect requests. STCP can further reduce the overhead caused by DOS attacks by avoiding processing SYN packets from a client that has been identified as sending an excess number of frivolous requests.

To optimize STCP’s ability to respond to DOS attacks, use the `detect_syn_attack` parameter. When the value of `detect_syn_attack` is `on`, STCP ignores requests from a client that sends an unreasonable number of failed connect requests over an extended time period, until the client reduces its rate of requests. The TCP protocol requires that a host respond to all connect requests by sending a reset reply packet (RST). In legitimate situations, a host typically sends RST packets if it is not listening to the target port. Setting the value of `detect_syn_attack` to `on` temporarily overrides STCP’s typical behavior for specific targeted sites, and enables STCP to log such activity and to identify the source of the packets.

By default, the value of `detect_syn_attack` is `off`. To set the value of `detect_syn_attack` to `on`, issue the following request of the `analyze_system` command:

```
as: set_stcp_param detect_syn_attack on
```

For information about STCP's ability to log information about breaches in security, see [“Logging Discarded Packets” on page 7-3](#), which includes an example of a message that STCP sends to the `syserr_log.date` file when the value of the `-detect_syn_attack` parameter is `on`.

## Logging Discarded Packets

By default, STCP logs information about attempted security breaches to the system error log file (`syserr_log.date`). Specifically, STCP logs information about the following discarded packets:

- LAND packets discarded
- connect requests discarded during shutdown of the STCP stack
- connect requests discarded when the value of the `-detect_syn_attack` parameter is `on`

The following is an example of a message that STCP sends to the `syserr_log.date` file when the value of the `-detect_syn_attack` parameter is `on`:

```
TCP: Excess connect failure rate 6/sec (dropping pkt 7) A2737601/C6938EF8:8535
```

STCP logs this information in such a way that excessive occurrences of these types of packets do not cause excessive log entries. Specifically, STCP reduces the number of log entries as it detects a higher volume of SYN packets; however, the messages show the total number of SYN packets discarded. You can also view the total number of SYN packets discarded by issuing the `list_stcp_params` request of the `analyze_system` command. In the output, the line `syms discarded` indicates the number of packets.

To control this logging feature, use the `log_syn_attacks` parameter. By default, the value of `log_syn_attacks` is `on`. To disable this logging feature, issue the following request of the `analyze_system` command:

```
as: set_stcp_param log_syn_attacks off
```

## Port Scanning

Another common technique of malicious users is port scanning. In *port scanning*, a client sends connect requests (that is, SYN packets) to all ports on a host in order to gain information about that host.

The TCP protocol requires a host to send an RST packet when it receives a connect request at a port that it is not listening to. When a client receives an RST packet, the client learns that the host is reachable and that the host is not listening to that port.

You can enable STCP to defend itself against port scanning by using the `prevent_port_scan` parameter. When `prevent_port_scan` is on, STCP does **not** send an RST packet when it receives a connect request at a port that it is not listening to. As a result, a user has a more difficult time obtaining this information from a Stratus module and using the information for malicious activity. However, when `prevent_port_scan` is on, STCP behavior becomes non-standard, and legitimate debugging activities may be more difficult.

By default, the value of `prevent_port_scan` is set to `off`. To set the value of `prevent_port_scan` to `on`, issue the following request of the `analyze_system` command:

```
as: set_stcp_param prevent_port_scan on
```

## Excerpt from `list_stcp_params` Output

The following excerpt from output of the `list_stcp_params` request of the `analyze_system` command shows the security-related STCP parameters.

```
as: list_stcp_params
```

```
STCP Parameters:
```

```
.  
  . [dots indicate output omitted]  
.  
log syn attacks [off/on]                (log_syn_attacks)    on  
.  
.  
.  
detect syn attack (off/on)              (detect_syn_attack)  off  
syms discarded                          0  
prevent port scanning (off/on)          (prevent_port_scan)  off  
.  
.  
.
```

For complete output of the `list_stcp_params` and `set_stcp_param` requests, see the *OpenVOS System Analysis Manual* (R073).

---

## Chapter 8

# STCP Networking Daemons

This chapter describes the STCP networking daemons, which are started as OpenVOS processes on the module. Once started, the daemon processes enable the module to handle incoming requests for STCP services. You must be logged in as a privileged user to start or stop most of the daemon processes. You must be logged in as the user `root` to start or stop the `inetd` daemon process. (For information about STREAMS daemons, see [“Starting STCP on the Module” on page 6-2.](#))

[Table 8-1](#) briefly describes the daemon processes; this chapter provides detailed descriptions in alphabetical order.

**Table 8-1. STCP Daemon Processes** *(Page 1 of 2)*

Daemon	Description
<code>bootpd</code>	This daemon process handles BOOTP client requests. It reads the <code>bootptab</code> database file. The <code>inetd</code> daemon process handles the initial client request for a BOOTP connection. The <code>inetd</code> daemon process then starts the <code>bootpd</code> daemon process under the user <code>nobody.nobody</code> .
<code>ftpd</code>	This daemon process is started by the <code>start_stcp.cm</code> command macro to handle requests from an FTP client.
<code>inetd</code>	This daemon process is started as <code>stcp_inetd</code> by the <code>start_inetd.cm</code> command macro to handle requests for <code>inetd</code> services. (The user <code>root</code> must issue the <code>start_inetd.cm</code> command macro.) This daemon process reads the <code>inetd.conf</code> database file, which identifies the sockets to create and the services to associate with the sockets. It then listens for requests for services received at the specified sockets. When <code>inetd</code> receives a connection request for a service, it invokes the appropriate service to handle the request. It supports only IPv4 addresses.
<code>ospfd</code>	This daemon process is started by the <code>start_stcp.cm</code> command macro to implement the OSPF routing protocol.
<code>rtsold</code>	This daemon process sends ICMPv6 Router Solicitation messages on specified interfaces.
<code>snmpd</code>	This daemon process is started by the <code>start_stcp.cm</code> command macro to support SNMP. It reads the <code>snmpconf</code> database file.

**Table 8-1. STCP Daemon Processes** (Page 2 of 2)

Daemon	Description
<code>tcpd</code>	This daemon process is started by the <code>inetd</code> daemon process to authenticate client requests for services that are specified in the <code>inetd.conf</code> file. (Client authentication is part of TCP wrappers functionality.) The <code>tcpd</code> daemon process checks the <code>hosts.allow</code> and <code>hosts.deny</code> database files for clients that can be allowed or denied access to various services.
<code>telnetd</code>	This daemon process is started by the <code>start_stcp.cm</code> command macro to handle login and slave STCP TELNET connections. In response to from connection requests a TELNET client, <code>telnetd</code> and the TLI software together establish TELNET sessions, pass input and output between a TELNET client and an OpenVOS process, and close the sessions. The <code>telnetd</code> daemon reads the <code>telnet-service</code> database file.
<code>tftpd</code>	This daemon process handles TFTP client requests. The <code>inetd</code> daemon process handles the initial client request for a TFTP connection. The <code>inetd</code> daemon process then forks off a new process for this request, and this new process starts the <code>tftpd</code> daemon process under the user <code>nobody.nobody</code> .
<code>tftpd6</code>	This daemon process handles TFTP client requests with support for IPv6 addresses.
<code>xinetd</code>	This daemon process listens for incoming requests for services on the module. It supports IPv6 addresses.

Once STCP is started and the daemon processes are running, they send any related error messages to their appropriate log files (for example, `inetd.out` or `bootplog`). If the module is rebooted, a new log file is generated for each daemon process, and the old log file is renamed to include a date and time stamp (for example, `inetd.03-10-26.21:16:56.out`). The directory `(master_disk)>system>stcp>logs` contains the generated log files. In some of the log files (for example, `ftpd.out` and `tftpdlog`), a date and time stamp is also appended to the error message.

To verify which daemon processes are currently running, issue the `list_users` command or the `netstat` command. The following example shows sample output from the `list_users` command.

```
list_users *.* -process *d -admin -system -interval
```

```

USERS at 10:01:03          CPU  DCPU      PF   DPF M  PS  F  P  PROGRAM
Overseer (ftpd)           0.0          65     0 WS + 7 ftpd.pm
Overseer (ospfd)          0.0          65     0 WS + 7 ospfd.pm
Overseer (snmpd)          0.0          61     0 WS + 7 snmpd.pm
Overseer (telnetd)        0.2          82     0 WS + 7 telnetd.pm
root (stcp_inetd)         0.0          79     0 WS + 7 inetd.pm

```

The following example shows sample output from the `netstat -all_sockets` command. Note that the output includes the trivial UDP- and TCP-based services available only through `inetd`, such as `echo`, `discard`, `daytime` (human-readable time), `chargen` (character generator), and `time` (machine-readable time).

```
netstat -all_sockets
```

```
Active connections (including servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp	0	0	*:echo	*:*	LISTEN
tcp	0	0	*:ftp	*:*	LISTEN
tcp	0	0	*:discard	*:*	LISTEN
tcp	0	0	*:daytime	*:*	LISTEN
tcp	0	0	*:chargen	*:*	LISTEN
tcp	0	0	*:time	*:*	LISTEN
tcp	0	0	*:telnet	*:*	LISTEN
tcp	0	0	172.16.2.38:telnet	france:2056	ESTABLISHED
udp	0	0	*:snmp	*:*	
udp	0	0	*:echo	*:*	
udp	0	0	*:discard	*:*	
udp	0	0	*:daytime	*:*	
udp	0	0	*:chargen	*:*	
udp	0	0	*:time	*:*	
udp	0	0	*:tftp	*:*	

## bootpd

**Privileged**

### Purpose

The `bootpd` daemon process handles boot requests from diskless devices on the network. Only the `inetd` daemon process can invoke `bootpd`.

### Display Form

There is no display form for this daemon process.

### Command-Line Form

You cannot start `bootpd` from command level.

### Explanation

The `bootpd` daemon supports the Bootstrap Protocol (BOOTP). The `inetd` daemon process invokes the `bootpd` daemon process in order to handle boot requests from diskless devices.

In order to handle boot requests, `bootpd` requires one entry in the `inetd.conf` database file, two entries in the `services` database file, and boot-related information in the `bootptab` database file. The two entries in the `services` database file show that `bootpd` is a UDP service associated with ports 67 and 68.

Typically, boot requests involve the following steps.

1. The `inetd` daemon process starts on the module, which enables the module to handle requests for its services from the network.
2. The device that needs information about itself as well as a boot file is activated and broadcasts a request using its `bootp` client.
3. The `inetd` daemon process responds by starting the `bootpd` daemon process as a separate daemon process, under the user `nobody.nobody`.
4. The `bootpd` daemon process checks the `bootptab` file for information about the device that issued the request.
5. The device learns that the information is available and invokes its `tftp` client to ask for the boot file.



6. The `inetd` daemon process starts the `tftpd` daemon process to handle the transfer of the file.

The `bootpd` daemon process writes messages to the `bootplog` log file in the directory `(master_disk)>system>stcp>logs`. Archived log files, which include a date and time stamp in the file name, also reside in this directory.

In order for the `bootpd` daemon to log messages to the `bootplog` log file, the STCP system administrator **must** give the user `nobody.nobody` modify access to the `(master_disk)>system>stcp>logs` directory. The STCP system administrator must also give the user `nobody.nobody` write access to the `bootplog` log files. The `bootpd` daemon process can write messages to the log file `bootplog` only when the user `nobody.nobody` has write access to the log file and modify access to the directory `(master_disk)>system>stcp>logs`.

The `bootpd` daemon process can use the `tcpd` daemon process for the authentication and logging features of TCP wrappers functionality.

#### NOTE

If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in BOOTP services.

## Examples

The following excerpt from the `inetd.conf` database file shows `tftpd` and `bootpd` entries. You must activate `tftpd` and `bootpd` entries in order to start the `bootpd` daemon process.

```
# Uncomment the following lines only if you want to use the bootp or tftpd server.
# You should be very careful since there are security issues with using tftp.
#
# tftpd    dgram  udp  wait  root  >system>stcp>command_library>tftpd.pm
# bootpd   dgram  udp  wait  root  >system>stcp>command_library>bootpd.pm
#
# If you are using tcpwrappers to control access to tftpd, use the following:
# tftpd    dgram  udp  wait   root  >system>stcp>command_library>tcpd.pm >system>s
+tcp>command_library>tftpd.pm
#
# If you are using tcpwrappers to control access to bootpd, use the following:
# bootpd   dgram  udp  wait   root  >system>stcp>command_library>tcpd.pm >system>s
+tcp>command_library>bootpd.pm
#
```

The following example is an excerpt from a `services` database file. This excerpt shows the entries required to handle `bootpd` and `tftpd` (`tftpd` handles the transfer of boot files).

<code>bootps</code>	<code>67/udp</code>	<code>bootpd</code>
<code>bootpc</code>	<code>68/udp</code>	<code>bootp</code>
<code>tftp</code>	<code>69/udp</code>	<code>tftpd</code>

## Related Information

See the description of the `tcpd` daemon process for information about using the process to authenticate and log BOOTP requests.

See [Chapter 5](#) for information about the `bootptab`, `inetd.conf`, and `services` database files.

# ftpd

**Privileged**

## Purpose

The `ftpd` daemon process implements the server side of the File Transfer Protocol (FTP). It receives and accommodates incoming connection requests from an FTP client.

## Display Form

```
----- ftpd -----
-debug:                no
-log:                  no
-syserr:               no
-security_check_file:
-ftp_port:             21
-allow_any_port:       no
-tcpwrapper_check:    no
-numeric:              no
-log_login_success:    yes
```

## Command-Line Form

```
ftpd [-debug]
      [-log]
      [-syserr]
      [-security_check_file security_file]
      [-ftp_port number]
      [-allow_any_port]
      [-tcpwrapper_check]
      [-numeric]
      [-no_log_login_success]
```

## Arguments

- `-debug` CYCLE  
Writes additional debugging information to the `ftpd.out` log file in the directory `(master_disk)>system>stcp>logs` when problems exist with the `ftpd` daemon process. By default (the value `no`), the additional debugging information

is not written to the log file. (In general, do **not** specify the `-debug` argument because it causes `ftpd` to generate large logs and to generate a log file for each child process. However, in some cases you may want to specify this option to enable the `ftpd` daemon process to aid in troubleshooting.)

- ▶ `-log` CYCLE  
Logs additional informational messages for each FTP session. By default (the value `no`), the additional informational messages for each session are not logged.
- ▶ `-syserr` CYCLE  
Writes all messages to the system error log file (`syserr_log.date`). By default (the value `no`), the `ftpd` daemon process sends the general server error messages to its log file (`ftpd.out`) in the `(master_disk)>system>stcp>logs` directory, but does not send messages to the `syserr_log.date` file.
- ▶ `-security_check_file security_file`  
Prevents unauthorized OpenVOS users from logging in to a remote host with FTP. If you also specify the `-log` argument, the `ftpd.out` log file records the following information.
  - if a user with null access attempts to log in via FTP
  - if the `ftpd` daemon process cannot find `security_file`

#### NOTE

Do not confuse the `-security_check_file` argument with the `-tcpwrapper_check` argument. The `-security_check_file` argument checks a valid OpenVOS user ID for access to FTP. The `-tcpwrapper_check` argument enables the `ftpd` daemon process to authenticate each client that requests an FTP connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files.

The value of `security_file` should be a path name to an empty file, which will become the security file. The access control lists for the security file control access to the system through an FTP connection. For example, if an OpenVOS user logging in through FTP has non-null access to the file, the user is granted access to the system and can successfully log in. If the user has null access to the file, the user receives an error message and the FTP connection is terminated.

You can create a list of allowed users by setting null access for `*.*` and giving read access for the specific users who are allowed access. You can create a list of denied users by giving read access to `*.*` and setting null access to the list of

specific users who are denied access. You can specify users as `user_name.group_name`, `user_name.*`, or `*.group_name`.

If the `ftpd` daemon process cannot find the security file, the user is denied access to the system.

If you want to start the `ftpd` daemon process with security enabled, issue the following command.

```
start_process -output_path (master_disk)>system>stcp>logs>ftpd.out
+'(master_disk)>system>stcp>command_library>ftpd -security_check_file
+%se_j14#m14>SW>Smith>ftp_security' -privileged -process_name ftpd7
```

#### NOTE

Do not enter the plus (+) sign that appears in this example; it is a line-continuation character.

#### ► `-ftp_port number`

Specifies the FTP port number. Values are valid numbers of TCP/IP ports that are not already in use. The default value is 21. If you specify a value less than 1024, you must issue `ftpd` as a privileged user.

#### ► `-allow_any_port`

CYCLE

Checks that the IP address in the FTP protocol command `PORT` matches the client's IP address. By default (the value `no`), the `ftpd` daemon process does not perform this check.

#### NOTE

The `ftpd` daemon process supports IPv4 and IPv6 addresses.

The `-allow_any_port` argument provides protection against security breaches by detecting the following information.

- whether the TCP/IP address in the `PORT` command is not from the originator
- whether the `PORT` number is less than 1024

You can use the `-allow_any_port` argument with the following external variables of the `ftpd` daemon.

- `allow_any_port$`—This variable is set to 1 if you specify the `-allow_any_port` argument. By default, it is set to 0. If you use proxy servers, you must set this variable or the proxy servers will not function properly.

- `warn_on_port_misuse$`—If this variable is set to 1 (the default) and a suspect `PORT` command is received when you are running the `ftpd` daemon process, the following message is returned.

```
'refused PORT (TCP/IP_address) from (client_name)'
```

- `warn_when_allow_port$`—If this variable is set to 1 (the default) and you specify the `-allow_any_port` argument, the following message is returned.

```
'Warning: ftpd invoked with -allow_any_port'
```

You can use the `set_external_variable` command to set the preceding external variables, as shown in the following example.

```
set_external_variable warn_on_port_misuse$ -in >system>stcp  
+>command_library>ftpd.pm -to 1 -type integer
```

#### NOTE

Do not enter the plus (+) sign that appears in this example; it is a line-continuation character.

#### ► `-tcpwrapper_check`

CYCLE

The value `yes` enables the `ftpd` daemon process to authenticate each client that requests an FTP connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files. As part of authenticating client requests, the daemon logs messages to the log files `tcpdallow` and `tcpddeny`. By default (the value `no`), the `ftpd` daemon process does not authenticate clients and does not log messages in the log files.

#### NOTE

Do not confuse the `-security_check_file` argument with the `-tcpwrapper_check` argument. The `-security_check_file` argument checks a valid OpenVOS user ID for access to FTP. The `-tcpwrapper_check` argument enables the `ftpd` daemon process to authenticate each client that requests an FTP connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files.

#### ► `-numeric`

CYCLE

The value `yes` enables the `ftpd` daemon process to check only the client's IP address and to ignore the host name when the daemon authenticates client access. By checking only the numeric IP address and ignoring the host name, the daemon can authenticate the client more quickly. By default (the value `no`), the

ftpd daemon process checks the client's IP address and the host name. The value that you specify for the `-numeric` argument takes effect only when the value of the `-tcpwrapper_check` argument is `yes`.

► `-no_log_login_success`

**CYCLE**

The value `yes` (the default) enables the ftpd daemon process to post messages of login success to the `ftpd.out` log file. These messages can cause the log file to become very large. To stop these messages, specify the value `no`.

## Explanation

This section summarizes important information concerning the ftpd daemon process.

### Starting ftpd

Start the ftpd daemon process from the `start_process` command line for the daemon in your `start_stcp.cm` command macro, which you must place in the `(master_disk)>system>stcp` directory. If you have to stop and restart the ftpd daemon process from command level, issue the `start_process` command to start ftpd as a privileged process. However, if you start the process from command level, you must issue additional commands that enable you to view the log file while the daemon is running. These commands include `create_file` and `set_implicit_locking`, as shown in the Examples section of this daemon description. (Due to the manner in which the ftpd daemon process is started in the `start_stcp.cm` command macro, you can view a log file while the daemon process is running.)

### Stopping ftpd

The `stop_process` command terminates the ftpd daemon process (as well as `stop_process` commands for the child processes); the process will not terminate on its own.

### Managing Child Processes

The ftpd daemon process, which must run as a privileged process, starts a uniquely named process called a *child process* for each session it establishes with a remote FTP client. The ftpd daemon process actually invokes the `ftpd_ch.pm` program module, which controls the creation of the child processes. Each child process created has a name that uses the format `ftpc.YY-MM-DD_HH:MM:SS_PPPPP` to identify the year, month, day, hour, minute, seconds, and the port number of the client (the port number accommodates up to five digits). A sample child process name is `ftpc.03-07-01_17:05:20_14869`.

### Specifying Database-file Information

To enable the ftpd daemon process to handle incoming connection requests, ensure that the `services` database file, in the `(master_disk)>system>stcp` directory, associates the FTP service with a TCP port number. The template `services` file, located in the `(master_disk)>system>stcp>templates` directory, has two

entries for FTP. The first entry associates the service `ftp-data` with port 20 and defines port 20 as a noninteractive FTP port. The second entry associates the service `ftp` with port 21 and defines port 21 as the interactive FTP command port.

## Log Files

By default, the `ftpd` daemon process sends the general server error messages to its log file in the `(master_disk)>system>stcp>logs` directory. The current (active) log file is named `ftpd.out`; if the daemon is stopped and restarted, a new `ftpd.out` file is created and the previous log file is archived using a date and time stamp (for example, `ftpd.03-05-15.07:35:07.out`). The error messages sent to this log file also contain a date and time stamp. Each child process also generates its own log file to contain any error messages associated with each FTP session. The log files generated by the child processes reside in the

`(master_disk)>system>stcp>logs` directory and are identified by their respective process names (for example, `ftpc.03-07-01_17:05:20_14869.out`). Child process log files are only created if there is a message to log. You should periodically delete the log files you no longer need. (Note that an additional file called `sftpc.out` will reside in the `logs` directory. This zero-length file must exist in the directory to ensure that the child processes can be spawned, but nothing will ever get written to it.)

You can also specify the `-syserr` argument, which enables the `ftpd` daemon process to send messages to the `syserr_log.date` file.

As part of authenticating client requests, the `ftpd` daemon process logs messages to the log files `tcpdallow` and `tcpddeny`. To enable authentication, specify the value `yes` for the `-tcpwrapper_check` argument.

In order for the `ftpd` daemon process to log messages to these log files, the STCP system administrator **must** give the user `nobody.nobody` modify access to the `(master_disk)>system>stcp>logs` directory. The STCP system administrator must also give the user `nobody.nobody` write access to the log files `tcpdallow`, `tcpddeny`, and `ftpd.out`. The `ftpd` daemon process can write messages to these files only when the user `nobody.nobody` has write access to the log files and modify access to the directory `(master_disk)>system>stcp>logs`.

The `ftpd` daemon logs information about RADIUS authentication to the `security_log.date` file (in the `(master_disk)>system` directory) when the OpenVOS RADIUS authentication server is running.

## Verifying ftpd

To verify that the `ftpd` daemon process is listening for requests, issue the `netstat` command with the `-all_sockets` argument. The output should include `ftp` listed with the state `LISTEN`. For an example, see Example 3 in the `netstat` command description.



## Authenticating Connections

The `ftpd` daemon process authenticates FTP users based on a user name and a password. In addition, the `ftpd` daemon process supports RADIUS authentication when the OpenVOS RADIUS authentication server is running, and it supports TCP wrappers authentication using the `-tcpwrapper_check` argument.

When the OpenVOS RADIUS authentication server is running, the `ftpd` daemon automatically determines if a user requires RADIUS authentication and logs information about it to the `security_log.date` file (in the `(master_disk)>system` directory).

If you specify the value `yes` for the `-tcpwrapper_check` argument, the `ftpd` daemon process authenticates each client that requests an FTP connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files.

## NOTES \_\_\_\_\_

1. If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in FTP services.
2. If the log files `tcpdallow` and `tcpddeny` become larger than 254 blocks, OpenVOS renames the files to `tcpdallow.yy-mm-dd.hh-mm` and `tcpddeny.yy-mm-dd.hh-mm` and starts new log files. (If you want to change the size at which log files become too large, you must contact the CAC.)

## Providing a Login Banner

The `ftpd` daemon process displays the text of an optional banner file when a client logs in to the daemon. The file name is `ftp_banner`, and it must reside in the `(master_disk)>system>stcp` directory. An administrator must create this file with zero or more lines of simple text.

## Examples

The following example identifies the FTP requests that are supported (and those that are unimplemented) by the STCP `ftpd` daemon process. The unimplemented requests are marked with an asterisk (\*). An FTP user issues the FTP subcommand `remotehelp` to display this list. (The number 214 indicates that this is a response from the server.)

214-The following commands are recognized (\* =>'s unimplemented).

USER	PASV	APPE	MRSQ*	ABOR*	SITE*	XMKD	XCUP
PASS	TYPE	MLFL*	MRCP*	DELE	SYST	RMD	STOU
ACCT*	STRU	MAIL*	ALLO	CWD	STAT	XRMD	SIZE
REIN*	MODE	MSND*	REST*	XCWD	HELP	PWD	RSIZ
QUIT	RETR	MSOM*	RNFR	LIST	NOOP	XPWD	MDTM
PORT	STOR	MSAM*	RNTO	NLST	MKD	CDUP	

Table 8-2 briefly describes the implemented requests in alphabetical order.

**Table 8-2. FTP Requests Supported by ftpd** (Page 1 of 2)

Request Supported by ftpd	Description
ALLO	Allocates storage
APPE	Appends a file
CDUP	Changes to the parent of the current working
CWD	Changes current working directory
DELE	Deletes a file
HELP	Provides help information
LIST	Lists files in a directory
MDTM	Displays date and time at which a specified file was last modified
MKD	Creates a directory
MODE	Specifies a data transfer mode
NLST	Provides a name list of files in a directory
NOOP	Does nothing
PASS	Specifies a password
PASV	Sets server in passive mode
PORT	Specifies a data connection port
PWD	Prints the current working directory
QUIT	Quits the session
RETR	Retrieves a file
RMD	Removes a directory
RNFR	Specifies rename-from file name

**Table 8-2. FTP Requests Supported by ftpd** (Page 2 of 2)

<b>Request Supported by ftpd</b>	<b>Description</b>
RNTO	Specifies rename-to file name
RSIZ	Sets fixed record size for type I
SIZE	Displays size, in bytes, of a file
STAT	Displays connection status
STOR	Stores a file
STOU	Stores a file with a unique name
STRU	Specifies data transfer structure
SYST	Displays system type
TYPE	Specifies data transfer type
USER	Specifies a user name
XCUP	Changes to the parent of current working directory
XCWD	Changes working directory
XMKD	Creates a directory
XPWD	Prints the current working directory
XRMD	Removes a directory

To start the daemon process from command level and to create the `ftpd.out` log file, issue the following command lines (as derived from the `start_stcp.cm` command macro).

```
create_file (master_disk)>system>stcp>logs>ftpd.out

set_implicit_locking (master_disk)>system>stcp>logs>ftpd.out

start_process -output_path (master_disk)>system>stcp>logs>ftpd.out
+ '(master_disk)>system>stcp>command_library>ftpd' -privileged
+ -process_name ftpd
```

## Log File Messages and the `-security_check_file` Argument

The following example shows what a user sees if the user is denied access to a system because the system administrator specified the `-security_check_file` argument.

```
ftp 127.0.0.1
Connected to 127.0.0.1.
220 medusa1 FTP server ready.
Name (127.0.0.1:Noah_Smith.User): noah.user
331 Password required for noah.user.
Password:

550 Noah_Smith.User not allowed to use FTP on this system.
Login failed.
ftp>
```

The following example shows the contents of the security log when a user is denied access to a system.

```
5: 99-06-12 15:19:21 MST Noah_Smith.User
    Event: SECURITY Status: FAILURE Process ID: 520181DA
    Target: FTP login failed from localhost
    Text: Noah_Smith.User not allowed to use FTP on this
        system.
```

The following example shows what a user sees if the user attempts to access a system and the security file is missing.

```
ftp 127.0.0.1
Connected to 127.0.0.1.
220 medusa1 FTP server ready.
Name (127.0.0.1:Noah_Smith.User):
331 Password required for Noah_Smith.User.
Password:

550 security_check file not found - no FTP logins are allowed.
Login failed.
```

The following example shows what the security log shows when the security file is missing.

```
6: 99-06-12 15:21:36 MST Noah_Smith.User
    Event: SECURITY Status: FAILURE Process ID: 520181DB
    Target: FTP login failed from localhost
    Text: Object not found.
security_check file se_j#m13_mas>SW>Noah_Smith>ftp_security
not found - no FTP logins are allowed.
```

In the preceding examples, note that the security log provides the path name of the missing security file while the message returned to the user does not. Also, note that the path name does not include the percent sign (%) character.

### Messages in the `tcpdallow` and `tcpddeny` Log Files

The following examples show entries that appear in the `tcpdallow` and `tcpddeny` log files (in the `(master_disk)>system>stcp>logs` directory) for `ftpd` connections.

- When the `ftpd` daemon process has authenticated and accepted a connection request, the following message appears in the `tcpdallow` log file.

```
01-01-29 16:15:01 est ftpd: connect from m3.company.com
```

- When the `ftpd` daemon process rejects a connection request because it was unable to authenticate it, the following message appears in the `tcpddeny` log file.

```
01-01-29 16:15:21 est ftpd: refused connect from m6.company.com
```

### Related Information

[“The `hosts.allow` and `hosts.deny` Files and TCP Wrappers” on page 5-10](#) describes the `hosts.allow` and `hosts.deny` files, and [“The `services` File and Service Information” on page 5-46](#) describes the `services` file.

[“Modifying and Executing the `start\_stcp.cm` Command Macro” on page 6-4](#) contains the `start_stcp.cm` command macro and the command lines for the `ftpd` daemon process.

[Chapter 9](#) contains the `netstat` command description.

RFC 765 provides a list of the FTP requests.

The manual *OpenVOS System Administration: Registration and Security* (R283) provides complete information about RADIUS authentication.

The *OpenVOS STREAMS TCP/IP User's Guide* (R421) describes FTP.

The *OpenVOS Commands Reference Manual* (R098) provides information about the `set_external_variable` and `stop_process` commands.

The *OpenVOS Commands User's Guide* (R089) provides information about standard access-control-list processing, which the `ftpd` daemon uses to determine access.

# inetd

## Privileged

### Purpose

The `inetd` daemon process listens for incoming requests for services on the module. The services invoked by `inetd` must be defined in the `inetd.conf` and `services` database files. You start the `inetd` daemon process by issuing the `start_inetd.cm` command macro as the user `root`.

#### NOTE

The `inetd` daemon process supports only IPv4 addresses. For IPv6 protocol support, use [xinetd](#).

### Display Form

```
----- inetd -----  
config: ██████████  
-debug_level: 0
```

### Command-Line Form

```
inetd [config] [-debug_level number]
```

### Arguments

- ▶ *config*  
The configuration file to be read for information about the STCP services that are available using the `inetd` daemon process on the module. By default, the daemon process looks for the file `(master_disk)>system>stcp>inetd.conf`.
- ▶ `-debug_level` CYCLE  
Controls the amount of information in log-file messages. By default, the `inetd` daemon process creates the log file `inetd.out` (in the `(master_disk)>system>stcp>logs` directory) and logs messages to it. Some child processes started by `inetd` also log messages to this file. The `-debug_level` argument enables you to control the amount of information in the messages sent to this file. The values are 0 and 1 (or any nonzero value).

The value 0 specifies that the daemon logs messages about the following events.

- services initially started
- services started when the `kill` command signals the `inetd` daemon process to reread the `inetd.conf` file.

The value 1 (or any nonzero value) specifies that the daemon logs messages about all of the events that are included with the value 0, and also the following events.

- connection requests
- process ID of a child process that it forks
- process ID of a child process that has completed its task
- additional events

For example, the following message, which indicates a request for the `swat` service though the `swat` service has not yet started, appears only when you specify the value 1.

```
Someone wants swat
```

## Explanation

A module requires the `inetd` daemon process when the module runs the `bootpd` daemon process, and when the module requires the services defined in the `inetd.conf` and `services` database files. Otherwise, basic functioning of STCP does not require the `inetd` daemon process.

The following list summarizes important information about the `inetd` daemon process.

- **Starting `inetd`**—Start the `inetd` daemon process from a `start_process` command line in the `start_inetd.cm` command macro, which you must place in the `(master_disk)>system>stcp` directory. You must be logged in as the user `root` to issue the `start_inetd.cm` command macro. The `start_stcp.cm` command macro invokes the `start_inetd.cm` command macro.
- **Process name**—When the `start_inetd.cm` command macro starts `inetd`, the STCP `inetd` daemon process is assigned the name `stcp_inetd`.
- **Log files**—When the `start_inetd.cm` command macro starts the `inetd` daemon process, the `inetd` log file, `inetd.out`, is created for log-file messages. You can check messages in this file even while the daemon process is running.
- **Managing services and other processes**—After it starts, the `inetd` daemon process reads a configuration database file (by default, `inetd.conf`), opens sockets for the services listed in that file, and then listens on those open sockets for incoming client connection requests. When it receives a connection request, the



inetd daemon process determines the corresponding service, examines the user field in the configuration database file, and invokes the appropriate daemon process to handle the request. The inetd daemon process continues to listen after the process has finished executing. The inetd daemon process handles minor internal, UDP- and TCP-based services such as `echo`, `discard`, `chargen` (character generator), `daytime` (human-readable time), and `time` (machine-readable time, in the form of number of seconds). Two external, UDP-based processes that you can configure inetd to invoke are `bootpd` and `tftpd`.

- Specifying `inetd.conf` file information—To enable the inetd daemon process to manage other processes, it must be able to find information in a configuration database file (by default, `inetd.conf`). The configuration file must reside in the `(master_disk)>system>stcp` directory. To use `bootpd` and `tftpd`, uncomment their command lines in the `inetd.conf` file.

#### NOTE

After you modify the `inetd.conf` database file, you must stop and restart the inetd daemon process to activate the changes, or you can activate those changes immediately by issuing the command `kill -s HUP pid`. (for the `pid` argument, specify the process ID of the inetd daemon process, which you can determine by issuing the command `lui -process_id`).

- Specifying `services` file information—All services managed by the inetd daemon process must have an entry in the `services` database file.
- Stopping inetd —The `stop_process` command terminates the inetd daemon process named `stcp_inetd`; the process will not terminate on its own. To stop `stcp_inetd`, simply issue the `stop_inetd.cm` command macro as a privileged user.

## Examples

The following example shows an excerpt from an `inetd.conf` database file.

```
# The parameters are in this order:
# service-name
# endpoint-type protocol
# wait-status
# uid
# server-program (full pathname)
# server-arguments
echo      stream tcp nowait root    internal  internal
echo      dgram  udp  wait  root    internal  internal
discard stream tcp nowait root    internal  internal
```

*(Continued on next page)*

```
discard dgram udp wait root internal internal
daytime stream tcp nowait root internal internal
daytime dgram udp wait root internal internal
chargen stream tcp nowait root internal internal
chargen dgram udp wait root internal internal
time stream tcp nowait root internal internal
time dgram udp wait root internal internal
#
# Uncomment the following lines only if you want to use the bootpd
# or the tftpd server. You should be very careful since there are security
# issues with using tftp.
# tftpd dgram udp wait nobody >system>stcp>command_library>tftpd.pm
# bootpd dgram udp wait nobody >system>stcp>command_library>bootpd.pm
#
# Uncomment the following line if you want to use the
# Samba Web Administration Tool (SWAT)
# swat stream tcp nowait root >system>samba>command_library>swat.pm
```

To start the daemon process from command level and to create the `inetd.out` log file, issue the following command lines (as derived from the `start_inetd.cm` command macro).

```
create_file (master_disk)>system>stcp>logs>inetd.out

set_implicit_locking (master_disk)>system>stcp>logs>inetd.out

start_process -output_path (master_disk)>system>stcp>logs>inetd.out
+'(master_disk)>system>stcp>command_library>inetd' -privileged
+-process_name stcp_inetd
```

## Related Information

[“The `inetd.conf` File and Information for `inetd`” on page 5-21](#) describes the `inetd.conf` database file and provides information about modifying it.

[“The `services` File and Service Information” on page 5-46](#) provides information about modifying the `services` file.

[Chapter 6](#) contains the `start_inetd.cm` and `stop_inetd.cm` command macros and the command lines for the `inetd` daemon process.

The *OpenVOS POSIX.1 Reference Guide* (R502) describes the `kill` command.

The *OpenVOS Commands Reference Manual* (R098) describes the `stop_process` command.

# ospfd

**Privileged**

## Purpose

The Open Shortest Path First (OSPF) daemon process, started by the `ospfd` command, enables an OpenVOS module to be a participant in an OSPF autonomous system (AS).

### NOTE \_\_\_\_\_

The `ospfd` daemon process supports only IPv4 addresses.

## Display Form

```
----- ospfd -----  
-config_file: >system>stcp>ospfdconf  
-event_log:  no
```

## Command-Line Form

```
ospfd [-config_file path_name] [-event_log]
```

## Arguments

- ▶ `-config_file path_name`  
Specifies the complete path name of the OSPF database file. If you do not specify a file name, the daemon uses the `ospfdconf` file, in the (master\_disk) >system>stcp directory, by default.
- ▶ `-event_log` CYCLE  
Displays event messages and saves them in the `ospfd.io` and `ospfd.out` files. By default, the event messages are not displayed and written to the `ospfd.out` file. (In general, do **not** enable the `-event_log` argument because it causes `ospfd` to constantly generate a large number of messages, which it saves in log files that become increasingly large.)

## Explanation

The `ospfd` daemon implements the OSPF link-state routing protocol for OpenVOS. OSPF is an interior router/gateway protocol that distributes routing information among participants in an *autonomous system* (AS), a collection of hosts and routers that are administered by a single organization. OSPF is also a link-state protocol, where each router (or module) maintains a *link-state database* (LSD) that describes the topology of the AS.

OSPF builds the LSD from the collective link-state advertisements (LSA) from all participants in the AS. A participant in the AS distributes its routing information to other participants by reliable flooding. The routing information it distributes consists of a list of usable interfaces and reachable neighbors.

OSPF constructs its LSAs from information in its OSPF configuration file, which, on an OpenVOS module, is named `ospfdconf`.

You can monitor modules and some other systems in an AS by running the `omon` command.

## Starting and Stopping the `ospfd` Daemon

Do not start the `ospfd` daemon process from the command line, except for debugging purposes. If you do start the daemon process from the command line, it displays verbose messages.

In general, you use the `start_stcp.cm` command macro to start `ospfd`, along with the other components of STCP. To enable the OSPF daemon, remove the ampersand (&) and space from OSPF entries in the `start_stcp.cm` file.

If necessary, you can start the `ospfd` daemon process within a `start_process` command line from OpenVOS command level. However, if the process is already started, you will get an error message.

To terminate the `ospfd` daemon process, use the `stop_process` command; the process will not terminate on its own.

## Example

The following example is an excerpt from a `start_stcp.cm` command macro. When you restart an OpenVOS module that contains a `start_stcp.cm` command macro with entries in this format, the command macro starts the `ospfd` daemon process.

```
& *****OSPFD*****
start_process -output_path
(master_disk)>system>stcp>logs>ospfd.out
+'(master_disk)>system>stcp>command_library>ospfd -config_file
+>system>stcp>ospfdconf' -privileged -process_name ospfd

!display_line OSPFD .....
```

## Related Information

“[The ospfdconf File and Routing Information](#)” on page 5-31 describes the `ospfdconf` file.

[Chapter 6](#) contains the `start_stcp.cm` command macro.

“[Problem 6: Module Cannot Route Packets](#)” on page 10-6 describes the debug log files.

The *OpenVOS Commands Reference Manual* (R098) describes the `start_process` and `stop_process` commands.

## **rtsold**

***Privileged***

### **Purpose**

The `rtsold` daemon process sends ICMPv6 Router Solicitation messages on specified interfaces.

### **Display Form**

```
----- rtsold -----  
interface:          █  
-autoprobe:         no  
-foreground:        no  
-configure:         no  
-single_probe:      no  
-mobile_node:       no  
-add_ra_src_address: no  
-pid_file:          no  
-other_script:       no  
-resolvconf_script: no  
-debug:             off
```

### **Command-Line Form**

```
rtsold [ interface(s) ]  
      [ -autoprobe ]  
      [ -foreground ]  
      [ -configure ]  
      [ -single_probe ]  
      [ -mobile_node ]  
      [ -add_ra_src_address ]  
      [ -pid_file PID_file_name ]  
      [ -other_script path_name ]  
      [ -resolvconf_script path_name ]  
      [ -debug string ]
```

## Arguments

- ▶ `interface(s)`  
The symbolic OpenVOS device name of one or more IPv6-capable SDLMUX-group interfaces configured on your module. You **must** precede the device name (or names) with the number sign (#) when specifying the network interface.
- ▶ `-autoprobe` CYCLE  
Enables the `rtsold` daemon process to scan all interfaces for those that are non-loopback, non-point-to-point, and are IPv6-enabled, and then the daemon process automatically operates on those interfaces. By default (the value `no`), the `rtsold` daemon process does not perform these actions.  
  
If the `ifconfig` command changes the configuration, the daemon process rescans the changed interfaces. The `-autoprobe` argument is mutually exclusive with the list of interfaces specified by the `interface` argument.
- ▶ `-foreground` CYCLE  
Specifies that `rtsold` executes as a command (in foreground mode). By default (the value `no`), `rtsold` executes as a daemon process.
- ▶ `-configure` CYCLE  
Explicitly configures the kernel to disable IPv6 forwarding and to accept router advertisements. By default (the value `no`), the `rtsold` daemon process does not perform these actions.
- ▶ `-single_probe` CYCLE  
The `rtsold` daemon process sends a single probe and then exits. By default (the value `no`), the `rtsold` daemon process sends multiple probes (for detailed information, see online BSD documentation such as <http://www.freebsd.org>).
- ▶ `-mobile_node` CYCLE  
Enables mobility support. The `rtsold` daemon process sends probing packets to default routers that have sent router advertisements when the node (re)attaches to an interface. By default (the value `no`), the `rtsold` daemon process does not send probing packets in this situation.
- ▶ `-add_ra_src_address` CYCLE  
Adds the source address of the router advertisement message to the interface name specified by the `-resolvconf_script` argument. By default (the value `no`), the `rtsold` daemon process does not add the address.
- ▶ `-pid_file PID_file_name`  
Specifies the name of the process ID (PID) file to which the `rtsold` daemon process writes its PID, instead of to the default file. By default, the daemon process writes its PID to `>system>stcp>rtsold.pid`.

- ▶ `-other_script path_name`  
Specifies the script to handle the Other Configuration Flag of the router advertisement.
- ▶ `-resolvconf_script path_name`  
Specifies the script that runs if the `rtsold` daemon process encounters the router advertisement options Recursive DNS Server (RDNSS) or DNS Search List (DNSSL).
- ▶ `-debug` CYCLE  
Enables debugging. Specify `on`, `off`, or `verbose`.

## Explanation

The `rtsold` daemon process sends ICMPv6 Router Solicitation messages on specified interfaces. The daemon process dynamically tracks and operates on non-loopback, non-point-to-point, and IPv6-capable interfaces. The daemon process also automatically tracks when interfaces come online or go offline.

Use the `rtsold` daemon process for all IPv6 interfaces that are not configured in a router.

## Related Information

See online BSD documentation (for example, <http://www.freebsd.org>).



## snmpd

## Privileged

### Purpose

The `snmpd` daemon process, a Simple Network Management Protocol (SNMP) daemon, enables remote network management stations that are running a third-party SNMP manager (client) to view and gather the SNMP statistics maintained by the STCP protocol drivers on the module.

#### NOTE

The `snmpd` daemon process supports only IPv4 addresses. For more complete SNMP support, including support of the IPv6 protocol, purchase the product EMANATE for OpenVOS, which is the Stratus OpenVOS product for Enhanced Management Agent Through Extensions (EMANATE®). For information, see *Software Release Bulletin: EMANATE for OpenVOS, Release 17.3 (R612)*.

### Display Form

There is no display form for `snmpd`.

### Command-Line Form

Start `snmpd` along with the other components of STCP using the `start_stcp.cm` command macro. If necessary, you can start the daemon process within a `start_process` command line from OpenVOS command level. However, if the process is already started, you will get an error message.

### Explanation

The following list summarizes important information about the `snmpd` daemon process.

- Variables—The STCP `snmpd` daemon process uses the UDP protocol and listens for SNMP requests on the defined `snmpd` port. As described in RFC 1157, the SNMP protocol allows the variables maintained by the SNMP daemon to be

inspected and altered by the remote network management stations. One important SNMP variable maintained by `snmpd` is `ipForwarding`, which determines whether the module can act as a router/gateway or simply as a host. The variables outlined in RFC 1213 are supported, except the EGP protocol group.

- **Communities**—The communities supported by `snmpd` are public, private, proxy, regional, and core, the null authentication scheme being used. The variables can be read by all communities. (The variables defined in RFC 1213 can only be written to by the private community.) You can change the strings that are used to refer to communities by entering community string values in the `snmpconf` file.
- **Specifying database-file information**—By default, the `snmpd` daemon process checks to see if the `snmpconf` database file exists. The `snmpconf` file can contain several site-specific system variables. Your version of this file must reside in the `(master_disk)>system>stcp` directory. The `snmpd` daemon process also requires an entry in the `services` database file, which must reside in the `(master_disk)>system>stcp` directory. The entry for this daemon process defines `snmpd` as a UDP service using the port number 161.
- **Starting `snmpd`**—Use the `start_stcp.cm` command macro to start the `snmpd` daemon process when you start STCP. However, to restart the daemon process from command level, issue the `start_process` command to start `snmpd` as a privileged process. (Note that if you start the daemon process from command level, you must issue the commands that enable you to view the current log file while the daemon process is running.)
- **Log files**—When the `start_stcp.cm` command macro starts the `snmpd` daemon process, the `snmpd` log file, `snmpd.out`, is created to hold any error messages generated by `snmpd`. You can check the errors in this file even while the process is running. If you use the `start_stcp.cm` command macro, all STCP log files, including `snmpd.out`, will be archived with a date and time stamp in the event that the module is rebooted. If no error occurs when you run the `snmpd` daemon process, it will run in the background and return immediately with code 0. If an error occurs, such as if the `services` database file does not contain an entry for `snmpd`, or if there is a problem finding or reading the `snmpconf` file, the error is written to the current `snmpd.out` file.

**Remotely settable IP variables**—As an SNMP daemon, `snmpd` maintains two variables that can be set by a remote SNMP manager whose community is set to private.

- **`ipForwarding`**—This SNMP IP variable (as described in RFC 2011) indicates whether the module with the SNMP daemon can act as a router/gateway or simply as a host. The value 1 means that the module can act as a router/gateway and forward datagrams received by, but not addressed to, this module; the value 2 means that the module is only a host and not a router/gateway (which means that a router/gateway on the network must forward packets). IP routers forward datagrams. By default, `ipForwarding` is set to 1 for STCP, thereby enabling the module to act as a router/gateway and

forward datagrams from one network to another when it has multiple connections to different networks/subnets. This variable can be manipulated by either a remote SNMP manager or by using the STCP command [IP\\_forwarding](#).

- `ipDefaultTTL`—This SNMP IP variable (as described in RFC 2011) is the Time-To-Live (TTL) default value inserted into the Time-To-Live field of the IP header of datagrams originated at this module, whenever a TTL value is not supplied by the transport layer protocol. The value can be 1 to 255.
- Stopping `snmpd`—The `stop_process` command terminates the `snmpd` daemon process; the daemon process will not terminate on its own.

## Examples

To start the `snmpd` daemon process from command level and to create the `snmpd.out` log file, issue the following command lines (as derived from the `start_stcp.cm` command macro).

```
create_file (master_disk)>system>stcp>logs>snmpd.out

set_implicit_locking (master_disk)>system>stcp>logs>snmpd.out

start_process -output_path (master_disk)>system>stcp>logs>snmpd.out
+ '(master_disk)>system>stcp>command_library>snmpd' -privileged
+-process_name snmpd
```

## Related Information

[“The `snmpconf` File and SNMP Information” on page 5-49](#) describes the `snmpconf` database file.

[Chapter 6](#) describes the `start_stcp.cm` command macro.

[Chapter 9](#) describes the [IP\\_forwarding](#) command.

The *OpenVOS Commands Reference Manual* (R098) describes the `stop_process` command.

## tcpd

## *Privileged*

### Purpose

As part of TCP wrappers functionality, the `tcpd` daemon process authenticates and logs client requests for services configured for it in the `inetd.conf` database file. The `inetd` daemon process typically starts the `tcpd` daemon process when entries in the `inetd.conf` database file specify `tcpd`.

#### NOTE

The `tcpd` daemon process supports both IPv4 and IPv6 protocols.

### Display Form

This daemon process has no display form.

### Command-Line Form

You cannot start `tcpd` from command level.

### Explanation

TCP wrappers functionality enables certain services (FTP, TELNET, and services that the `inetd` daemon process starts) to authenticate and log client requests. Services that the `inetd` daemon process starts (for example, BOOTP and TFTP) require the `tcpd` daemon for authentication and logging functionality. The `inetd` daemon process typically starts the `tcpd` daemon process when `tcpd` is included in an entry for a requested service in the `inetd.conf` database file.

When an entry for a service in the `inetd.conf` database file includes `tcpd`, the `inetd` daemon process starts the `tcpd` daemon process. The `tcpd` daemon process then authenticates and logs the client request. The `tcpd` daemon process checks the `hosts.allow` and `hosts.deny` database files. If the `tcpd` daemon process finds the client listed in the `hosts.allow` file with the appropriate service daemon, it accepts the connection request and logs a message in the `tcpdallow` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the client connected to the server.

If the `tcpd` daemon process does not find the client listed in the `hosts.allow` file with the appropriate service daemon, it searches the `hosts.deny` file. If the `tcpd` daemon process finds the client listed in the `hosts.deny` file with the appropriate service daemon, it denies the client access—if the client sent a TCP request, the `tcpd` daemon process closes the connection; or if the client sent a UDP request, the packet is dropped. The `tcpd` daemon process logs a message to the `tcpddeny` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the connection was refused.

If the client is not listed in either the `hosts.allow` file or the `hosts.deny` file, the `tcpd` daemon process accepts the connection and logs a message to the `tcpdallow` log file (in the directory `(master_disk)>system>stcp>logs`) stating that the client connected to the server. By default, the daemon accepts connections.

The following examples show entries that appear in the `tcpdallow` and `tcpddeny` log files for client connection requests that the `tcpd` daemon process has accepted or rejected.

- When the `tcpd` daemon process has authenticated and accepted a connection request, the following message appears in the `tcpdallow` log file.

```
01-01-29 16:14:24 est tcpd: connect from m1.company.com for tftpd.pm
```

- When the `tcpd` daemon process rejects a connection request because it was unable to authenticate it, the following message appears in the `tcpddeny` log file.

```
01-01-29 16:14:28 est tcpd: refused connect from m4.company.com for tftpd.pm
```

If the log files `tcpdallow` and `tcpddeny` become larger than 256 blocks, OpenVOS renames the files to `tcpdallow.yy-mm-dd.hh-mm` and `tcpddeny.yy-mm-dd.hh-mm` and starts new log files. (If you want to change the size at which log files become too large, you must contact the CAC.)

## Related Information

See the description of the [inetd](#) daemon process earlier in this chapter.

“[The hosts.allow and hosts.deny Files and TCP Wrappers](#)” on page 5-10 describes the `hosts.allow` and `hosts.deny` database files.

“[The inetd.conf File and Information for inetd](#)” on page 5-21 describes the `inetd.conf` database file and contains entries in the `inetd.conf` database file that invoke the `tcpd` daemon process.

## telnetd

**Privileged**

### Purpose

The `telnetd` daemon process implements the server side of the TELNET virtual terminal protocol. It receives and accommodates incoming connection requests from TELNET clients and can initiate TELNET connection requests to clients. In response to a client request, the `telnetd` daemon process and the TLI software establish TELNET sessions, pass input and output between a TELNET client and an OpenVOS process, and close the TELNET sessions.

### NOTES

---

1. The standard STCP TELNET server (started by the `telnetd` command) does not support the IPv6 protocol; instead, it supports only IPv4. Moreover, the TELNET protocol is non-secure. To increase security and/or to accommodate incoming login requests and incoming slave requests with IPv6 protocol support, use the secure shell (SSH) utility. For information on SSH, see the *Software Release Bulletin: Internet Security Pack for OpenVOS, Release 4.0.2 (R660)*.
2. STCP supports two implementations of TELNET: the standard STCP TELNET server (started by the `telnetd` command) and the Multisession TELNET (MST) server (started by the `telnet_msd` command). The MST server is for legacy applications only; for information on it, see [Appendix F, "The MST Server."](#) All references to TELNET in this manual are to the standard STCP TELNET server, unless otherwise noted.

### Display Form

```
----- telnetd -----  
-service_file:      >system>stcp>telnetSERVICE  
-tcpwrapper_check: no  
-numeric:          yes
```

## Command-Line Form

```
telnetd [-service_file path_name]
        [-tcpwrapper_check]
        [-no_numeric]
```

## Arguments

- ▶ `-service_file path_name`  
Specifies the complete path name of the file defining the local TELNET services on the module. By default, the path name of the file defining local TELNET services is `(master_disk)>system>stcp>telnetSERVICE`.
- ▶ `-tcpwrapper_check` CYCLE  
Enables the `telnetd` daemon process to authenticate each client that requests a TELNET connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files. As part of authenticating client requests, the daemon logs messages to the log files `tcpdallow` and `tcpddeny`. To enable authentication, specify the value `yes`. By default (the value `no`), the `telnetd` daemon process does not authenticate clients and does not log messages in the log files.
- ▶ `-no_numeric` CYCLE  
Enables the `telnetd` daemon process to check only the client's IP address and to ignore the host name when the daemon authenticates client access. By checking only the numeric IP address and ignoring the host name, the daemon can authenticate the client more quickly. By default (the value `yes`), the `telnetd` daemon process checks only the client's IP address. The value that you specify for the `-numeric` argument takes effect only when the value of the `-tcpwrapper_check` argument is `yes`.  
  
To enable the daemon to check the host name, specify the value `no` for the `-numeric` argument when you specify the value `yes` for the `-tcpwrapper_check` argument. If you specify the value `no` for the `-tcpwrapper_check` argument, the `-numeric` argument has no effect.

## Explanation

The following list summarizes important information concerning the `telnetd` daemon process.

- **Basic functioning**—The `telnetd` daemon process listens for connections on the defined TELNET ports. When a TELNET session starts, the daemon sends a TELNET option to the client side to indicate a willingness to perform remote echoing of characters. The daemon process supports binary mode and most of the common TELNET options, except timing marks.

- Specifying `telnet` service file information—By default, the `telnetd` daemon process checks the TELNET information in the `telnet` service database file. To define more than the standard login service, create additional service entries in your own `telnet` service file (for example, to handle incoming slave connections or additional login services that may use the `linger` option). Your final version of the `telnet` service database file must reside in the `(master_disk)>system>stcp` directory.
- Specifying `services` file information—The `telnetd` daemon process also requires an entry for the standard login TELNET service and an entry for each additional incoming service in the `services` database file, which must reside in the `(master_disk)>system>stcp` directory. The standard TELNET service defines `telnetd` as a TCP service using the port number 23.
- Device entries—Each TELNET service recognized by `telnetd` must be associated with at least one device in the `devices.tin` file, so you must create one or more device entries with the appropriate information for each TELNET login, incoming slave, or outgoing slave service.
- Dynamic changes—If you need to change, add, delete, or check a local TELNET service while the `telnetd` daemon process is running, issue the `telnet_admin` command. This command automatically updates the `telnet` service and `services` database files.
- Starting `telnetd`—Use the `start_stcp.cm` command macro to start the `telnetd` daemon process. If, however, you have to stop and then restart the process from command level, you can issue the `start_process` command to start `telnetd` as a privileged process. (Note that if you start the process from command level, you must also issue the commands that enable you to view the log file while the daemon process is running.)
- Log files (`telnet.out`, `tcpdallow`, and `tcpddeny`)—When the `start_stcp.cm` command macro starts the `telnetd` daemon process, the `telnetd` log file, `telnetd.out`, is created to hold the error messages. You can check the errors in this file even while the daemon process is running. If you use the `start_stcp.cm` command macro, OpenVOS archives all STCP log files, including `telnetd.out`, with a date and time stamp in the event that the module is rebooted.

As part of authenticating client requests, the daemon logs messages to the log files `tcpdallow` and `tcpddeny`. To enable authentication, specify the value `yes` for the `-tcpwrapper_check` argument.

The `telnetd` daemon logs information about RADIUS authentication to the `security_log.date` file (in the `(master_disk)>system` directory) when the OpenVOS RADIUS authentication server is running.

- Keepalive, no-delay, and `linger`—STCP enables the keepalive and no-delay options for the default TELNET login service defined in the template



`telnet` service database file. Consider specifying the `keepalive` and `no-delay` options for all of your TELNET services. You can also specify the `linger` option for a TELNET service in the local `telnet` service file or with the `telnet_admin` command.

- **Verifying `telnetd`**—To verify that the `telnetd` daemon process is listening for requests, issue the `netstat` command with the `-all_sockets` argument.
- **Stopping `telnetd`**—The `stop_process` command terminates the `telnetd` daemon process. You typically stop `telnetd` only when you stop STCP.
- **Authenticating client requests**—The `telnetd` daemon process supports RADIUS authentication when the OpenVOS RADIUS authentication server is running. It also supports TCP wrappers authentication.

When the OpenVOS RADIUS authentication server is running, the `telnetd` daemon automatically determines if a user requires RADIUS authentication and logs information about it to the `security_log.date` file (in the `(master_disk)>system` directory).

If you specify the value `yes` for the `-tcpwrapper_check` argument, the `telnetd` daemon process authenticates each client that requests a TELNET connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files.

## NOTES \_\_\_\_\_

1. If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in TELNET services.
2. If the log files `tcpdallow` and `tcpddeny` become larger than 256 blocks, OpenVOS renames the files to `tcpdallow.yy-mm-dd.hh-mm` and `tcpddeny.yy-mm-dd.hh-mm` and starts new log files. (If you want to change the size at which log files become too large, you must contact the CAC.)

## Examples

This section provides examples that show important information for `telnetd`.

### Example 1

To start the process from command level and to create the `telnetd.out` log file, issue the following command lines (as derived from the `start_stcp.cm` command macro).

```
create_file (master_disk)>system>stcp>logs>telnetd.out

set_implicit_locking (master_disk)>system>stcp>logs>telnetd.out

start_process -output_path (master_disk)>system>stcp>logs>telnetd.out
+'(master_disk)>system>stcp>command_library>telnetd' -privileged
+-process_name telnetd
```

### Example 2

The following example shows sample entries for TELNET services in the `telnetSERVICE` database file used by `telnetd`.

```
telnet      window_term "keepalive nodelay"      "Default Telnet login"  1 1 tli_logm1
m1slave1    window_term "keepalive nodelay"      "slave1 service"       0 0 tli_inslv1m1
m1slave2    window_term "keepalive linger=10 nodelay" "slave2 linger"       0 0 tli_inslv2m1
m1slave3    window_term "keepalive linger=15 nodelay" "slave3 linger"       0 0 tli_inslv3m1
```

### Example 3

The following example shows sample entries for TELNET services in the `SERVICES` database file used by `telnetd`.

```
telnet      23/tcp      telnetd    # default TELNET port
m1slave1    950/tcp      # inslave 1
m1slave2    951/tcp      # inslave 2 linger
m1slave3    952/tcp      # inslave 3 linger
```

### Example 4

The following examples show entries that appear in the `tcpdallow` and `tcpddeny` log files (in the `(master_disk)>system>stcp>logs` directory) for `telnetd` connections.

- When the `telnetd` daemon process has authenticated and accepted a connection request, the following message appears in the `tcpdallow` log file.
- When the `telnetd` daemon process rejects a connection request because it was unable to authenticate it, the following message appears in the `tcpddeny` log file.

```
01-01-29 16:14:36 est telnetd: connect from m2.company.com
```

```
01-01-29 16:14:41 est telnetd: refused connect from m5.company.com
```

## Related Information

“[Defining STCP TELNET Devices](#)” on page 4-8 describes how to create TELNET device entries.

“[The `hosts.allow` and `hosts.deny` Files and TCP Wrappers](#)” on page 5-10 describes the `hosts.allow` and `hosts.deny` database files.

“[The `services` File and Service Information](#)” on page 5-46 describes the `services` database file.

“[The `telnet.service` File and TELNET Information](#)” on page 5-53 describes the `telnet.service` database file.

[Chapter 6](#) describes the `start_stcp.cm` file.

[Chapter 9](#) describes the `netstat` and `telnet_admin` commands.

“[STCP Options](#)” on page A-21 describes the `keepalive`, `no-delay`, and `linger` options.

The manual *OpenVOS System Administration: Registration and Security* (R283) provides complete information about RADIUS authentication.

The *OpenVOS Commands Reference Manual* (R098) describes the `stop_process` command.

## tftpd

## Privileged

### Purpose

The `tftpd` daemon process implements the server side of the Trivial File Transfer Protocol (TFTP). Only the `inetd` daemon process can invoke `tftpd`.

The `tftpd` daemon process has no display form or command-line form because it is invoked by the `inetd` daemon process; however, the `tftpd` daemon process has one argument, `-logging`.

#### NOTE

The `tftpd` daemon process does not support the IPv6 protocol. Use `tftpd6` for IPv6 protocol support.

### Display Form

There is no display form for this daemon process.

### Command-Line Form

You cannot invoke `tftpd` from command level.

### Arguments

#### ► `-logging`

Specifies that the `tftpd` daemon process logs messages about TFTP transactions to the `tftpdlog` log file (in the directory `(master_disk)>system>stcp>logs`). By default, the `tftpd` daemon process does not log messages about TFTP transactions in the log file.

### Explanation

The `inetd` daemon process invokes the `tftpd` daemon process in order to handle TFTP requests (such as boot-file related requests) from a TFTP client. The `tftpd` daemon process handles multiple clients simultaneously. Specifically, the `tftpd` daemon functions as a concurrent User Datagram Protocol (UDP) server, which means that it can simultaneously accept TFTP requests from multiple clients. The `inetd`

daemon process handles the initial client request for a TFTP connection. The `inetd` daemon process then forks off a new process for this request, and this new process executes the `tftpd.pm` program module to run the `tftpd` daemon under the user `nobody.nobody`.

The `tftpd` daemon process makes log-file entries to the log file `tftplog` in the directory `(master_disk)>system>stcp>logs`.

When a client writes a file to the server without specifying a directory for it, the `tftp` daemon places the file in the `(master_disk)>system>stcp>tftp_default` directory.

The STCP system administrator **must** give the user `nobody.nobody` modify access to the following directories.

- `(master_disk)>system>stcp>tftp_default` (if clients need to write files to this directory on the server) —The `tftpd` daemon enables clients to read files in and write files to directories on the server only when the user `nobody.nobody` has proper access. Clients can read only the files to which the user `nobody.nobody` has read access. Clients can write files only to the directories to which the user `nobody.nobody` has modify access. When a client writes a file to the server but does not specify a directory for the file, the `tftp` daemon places that file in the directory `(master_disk)>system>stcp>tftp_default`, if the user `nobody.nobody` has modify access to it.
- `(master_disk)>system>stcp>logs`, which includes the `tftplog` log file—In order for the `tftpd` daemon to make log-file entries to the `tftplog` log file, the STCP system administrator **must** give the user `nobody.nobody` modify access to the `(master_disk)>system>stcp>logs` directory. The STCP system administrator must also give the user `nobody.nobody` write access to the `tftplog` file.
- `(master_disk)>system>stcp>firmware_cpcnnnn` directories, so that Intelligent Shelf Module (ISM) Controllers in a network I/O (NIO) enclosure can transfer files to a system (for systems that contain a NIO). For the exact directory names, see the *Stratus ftServer: Network I/O Enclosure Guide* (R608).

The `tftpd` daemon process requires an activated entry in the `inetd.conf` database file and an entry in the `services` database file. Both of these files reside in the `(master_disk)>system>stcp` directory. The entry for the `tftpd` daemon in the `services` file defines `tftpd` as a UDP service using the port number 69.

## NOTICE

Before activating the `inetd.conf` entry for `tftpd`, be aware of the serious security implications. No security is built into the TFTP software.

Use the `stop_process` command to terminate the `tftpd` daemon process.

The `tftpd` daemon process will allow only publicly readable files to be read and publicly writable files to be written (in this case, public includes all users on all hosts that can be reached through the network, which may not be appropriate on all systems). The `tftp` client does not require an account or password on the remote system.

The `tftpd` daemon process requires the `tcpd` daemon process for the authentication and logging features of TCP wrappers functionality.

#### NOTE

If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in TFTP services.

## Examples

The following excerpt from an `inetd.conf` database file shows entries for `tftpd` and `bootpd`. You must activate an `tftpd` entry in order to start the `tftpd` daemon process.

```
# Uncomment the following lines only if you want to use the bootp
# or the tftpd server.
# You should be very careful since there are security issues with
# using tftp.
# tftpd  dgram  udp  wait  root  >system>stcp>command_library>tftpd.pm
# bootpd dgram  udp  wait  root  >system>stcp>command_library>bootpd.pm
#
# If you are using tcpwrappers to control access to tftpd, use the following:
# tftpd  dgram  udp  wait  root  >system>stcp>command_library>tcpd.pm >system>s
+tcp>command_library>tftpd.pm
#
# If you are using tcpwrappers to control access to bootpd, use the following:
# bootpd dgram  udp  wait  root  >system>stcp>command_library>tcpd.pm >system>s
+tcp>command_library>bootpd.pm
#
```

The following example is an excerpt from a `services` database file. This excerpt shows the entries required to handle the `tftpd` and `bootpd` services (`bootpd` relies on `tftpd` for handling boot files).

<code>bootps</code>	<code>67/udp</code>	<code>bootpd</code>
<code>bootpc</code>	<code>68/udp</code>	<code>bootp</code>
<code>tftp</code>	<code>69/udp</code>	<code>tftpd</code>

## Related Information

See the description of the `tcpd` daemon process for information about using it to authenticate and log TFTP requests. See [“The `inetd.conf` File and Information for `inetd`” on page 5-21](#) for information about the `inetd.conf` database file. See [“The `services` File and Service Information” on page 5-46](#) for information about the `services` database file.

## tftpd6

***Privileged***

### Purpose

The tftpd6 daemon process implements the server side of the Trivial File Transfer Protocol (TFTP), with support for IPv6 addresses. Only the [xinetd](#) daemon process can invoke tftpd6.

The tftpd6 daemon process has no display form or command-line form because it is invoked by the [xinetd](#) daemon process; however, the tftpd6 daemon process has one argument, `-logging`.

### Display Form

There is no display form for this daemon process.

### Command-Line Form

You cannot invoke tftpd6 from command level.

### Arguments

► `-logging`

Specifies that the tftpd6 daemon process logs messages to the system error log file (`syserr_log.date`). By default, the tftpd6 daemon process does not log messages in the log file.

### Explanation

The tftpd6 daemon process supports IPv6 addresses, and, if you specify the `-logging` argument, logs messages to the system error log file. Otherwise, the tftpd6 daemon process is identical to the [tftpd](#) daemon process. For information, see the description of the [tftpd](#) daemon process.



## xinetd

## *Privileged*

The `xinetd` daemon process listens for incoming requests for services on the module.

### NOTE

The `xinetd` daemon process supports IPv6 addresses.

To start the `xinetd` daemon process, specify the value `yes` for the `-xinetd` argument of the `start_stcp.cm` command macro. For more information about starting the daemon process, see “[Starting the inetd or xinetd Daemon Process](#)” on page 6-9.

For information about defining services that you want the `xinetd` daemon process to start, see “[The xinetd.conf File and Information for xinetd](#)” on page 5-57.

For information on `xinetd` arguments, see <http://linux.die.net/man/8/xinetd>.



---

## Chapter 9

# STCP Commands

This chapter describes the following STCP commands.

- `access_info_monitor`
- `arp`
- `dlmux_admin`
- `ftp`
- `hostname`
- `ifconfig`
- `IP_forwarding`
- `ip_pair_admin`
- `ip6addrctl`
- `netstat`
- `ngrep`
- `omon`
- `packet_monitor`
- `ping`
- `route`
- `rtsol`
- `set_udp_ordering`
- `setkey`
- `tcpdump`
- `telnet`
- `telnet_admin`
- `tftp`
- `tftp6`
- `traceroute`

These STCP commands enable you to manage your STCP configuration and perform some common operations. If a command is labeled as privileged, you must be a privileged user to issue the command. Although commands such as `arp`, `netstat`, and `ping` can be issued by general users, they are typically of more use to the system administrator and are therefore documented in this manual only. General-use commands such as `ftp` and `telnet` are included in this manual for quick-reference purposes only; for a tutorial on `ftp` and `telnet`, see the *OpenVOS STREAMS TCP/IP User's Guide* (R421).

Although the `(master_disk)>system>stcp>command_library` directory includes `start_stcp_stack`, this command is not intended for general use; it is available only to Stratus support personnel. Administrators must use the `start_stcp.cm` command macro, which invoke these commands.

[Table 9-1](#) briefly describes the commands in alphabetical order. The remainder of this chapter provides detailed command descriptions that appear in alphabetical order.

**Table 9-1. STCP Commands** (Page 1 of 2)

Commands	Description
<code>access_info_monitor</code>	Enables you to display information about or to change the value of variables that are in the kernel and are associated with the SNMP object identifiers (OIDs).
<code>arp</code>	Enables you to list, delete, or set an entry in the translation table used by the Address Resolution Protocol (ARP), which converts an IP address to the appropriate MAC address (for example, a 48-bit Ethernet address).
<code>dlmux_admin</code>	Enables you to manage SDLMUX-associated partnered network interfaces.
<code>ftp</code>	Enables general users to perform file-transfer operations to a remote host.
<code>hostname</code>	Sets and prints the name of the current host module.
<code>ifconfig</code>	Enables you to add, delete, or check the status of a network interface during the current bootload. You can also change the state of a configured network interface (up or down) and add an additional IP address to a network interface.
<code>IP_forwarding</code>	Displays or changes the setting for the IPv4 forwarding feature. By default, IP forwarding is enabled, which allows the module to act as a router/gateway when it has network interfaces connected to different subnets.
<code>ip_pair_admin</code>	Enables you to control path MTU discovery on a network path between two IP addresses.

**Table 9-1. STCP Commands** (Page 2 of 2)

Commands	Description
<code>ip6addrctl</code>	Configures the IPv4 and IPv6 address selection policy.
<code>netstat</code>	Displays information about the status of the network, such as the status of network interfaces, the status of connections, and statistics pertaining to packet traffic. The information displayed, and the format in which it is presented, depends on the command arguments that you specify.
<code>omon</code>	Starts an OSPF monitoring session that allows you to manage OSPF routers that support the <code>omon</code> command and to obtain information from log files on those routers.
<code>packet_monitor</code>	Displays information about packets transmitted over the LAN through network interfaces configured on your module. You can display information about selected packets by filtering the packets that you want to monitor.
<code>ping</code>	Transmits an ICMP <code>ECHO_REQUEST</code> packet (for IPv4) or various ICMPv6 queries (for IPv6) to a specified remote host to determine whether the remote host is available on the network.
<code>route</code>	Enables you to alter network routing tables. The routing tables contain routing information about remote and local IP nodes. The <code>route</code> command also allows you to display routes.
<code>rtsol</code>	Invokes the <code>rtssold</code> daemon process.
<code>set_udp_ordering</code>	Enables you to specify the type of ordering of UDP datagrams for all network components.
<code>setkey</code>	Adds, updates, dumps, or flushes Security Association Database (SAD) and Security Policy Database (SPD) entries in the kernel.
<code>telnet</code>	Enables general users to perform remote logins to other hosts.
<code>telnet_admin</code>	Enables you to add, delete, modify, or verify a TELNET service on the module while the <code>telnetd</code> daemon process is running.
<code>tftp</code>	Enables general users to perform a file-transfer operation without user authentication. This utility is similar to FTP.
<code>tftp6</code>	Provides <code>tftp</code> functionality with support for IPv6 addresses.
<code>traceroute</code>	Enables you to trace the route that an IP packet follows to the specified network host.

## access\_info\_monitor

***Privileged***

### Purpose

The `access_info_monitor` command enables you to display information about or change the value of variables that are in the kernel and are associated with SNMP or OpenVOS object identifiers (OIDs).

In order to invoke this command, you must include the `>system>maint_library` directory (the directory in which the `access_info_monitor.pm` file resides) in the command library path for your user login.

### Display Form

```
----- access_info_monitor -----
oid:
command:          walk
value:
-module:
-exclude_std_oids:  no
-exclude_vos_oids:  no
-include_titles:    yes
-include_snmp_syntax: yes
-return_many:       yes
-return_templates:  no
-asnl_syntax:       no
```

## Command-Line Form

```
access_info_monitor [oid]
                    [command]
                    [value]
                    [-module module_name]
                    [-exclude_std_oids]
                    [-exclude_vos_oids]
                    [-no_include_titles]
                    [-no_include_snmp_syntax]
                    [-no_return_many]
                    [-return_templates]
                    [-asn1_syntax]
```

## Arguments

### ► *oid*

The SNMP OID that locates the kernel variable. If the value of the *command* argument is *set*, you must specify a scalar variable for the *oid* argument; otherwise, specify a scalar, row or table. By default (specifying no value for *oid*), the command output includes all kernel variables accessible through this interface.

### ► *command*

CYCLE

Specify one of the following:

- *walk* (the default)—The *access\_info\_monitor* command displays the kernel variable(s).
- *set*—The *access\_info\_monitor* command sets the kernel variable located by *oid* to the string specified by the *value* argument. The value *set* is supported for only some variables and is for privileged users only.

### ► *value*

Sets *oid* to *value* when the value of the *command* argument is *set*. If the value of the *command* argument is *walk*, *value* is ignored. If the string you specify for *value* includes punctuation or one or more spaces, enclose the string within apostrophes ( ' ' ), as in the following example:

```
access_info_monitor 1.3.6.1.2.1.1.5 set '%s#m1'
```

### ► -module

The module on which you are monitoring OIDs.

### ► -exclude\_std\_oids

CYCLE

Excludes standard OIDs from the output of the *access\_info\_monitor* command, when the value of the *command* argument is *walk*. By default (the

value `no`), the output of the `access_info_monitor` command displays all standard OIDs.

- ▶ `-exclude_vos_oids` CYCLE  
Excludes OpenVOS extensions from the output of the `access_info_monitor` command, when the value of the `command` argument is `walk`. By default (the value `no`), the output of the `access_info_monitor` command displays all OpenVOS extensions.
- ▶ `-include_titles` CYCLE  
By default (the value `yes`), the output of the `access_info_monitor` command displays mnemonic titles (that is, variable names), when the value of the `command` argument is `walk`. Specify the value `no` to exclude mnemonic titles from the command output.
- ▶ `-include_snmp_syntax` CYCLE  
By default (the value `yes`), the output of the `access_info_monitor` command displays the SNMP syntax (that is, data type) with each variable, when the value of the `command` argument is `walk`. Specify the value `no` to exclude SNMP syntax from the command output.
- ▶ `-return_many` CYCLE  
By default (the value `yes`), the `access_info_monitor` command retrieves many objects from the kernel at one time, when the value of the `command` argument is `walk`. Specify the value `no` to retrieve only one object at a time, which is useful when checking for bugs in the kernel code. The output format is the same, with `yes` or `no`.
- ▶ `-return_templates` CYCLE  
Specifies that the command output lists all of the OIDs and variable values contained in the OpenVOS MIB files, even if they are in an empty table. The command output lists just one repetition for each table. By default (the value `no`), the command output does not list empty tables.

For information about the OpenVOS MIB files, see [Table 9-7](#).

- ▶ `-asn1_syntax` CYCLE  
Specifies that the command output lists all OIDs, including and below the OID you specify for the `oid` argument in abstract syntax notation one format (that is, ASN.1 format). (ASN.1 format is the standard format for Internet MIBs.) When you specify this argument, you must specify a value for the `oid` argument that is a row with two parents. By default (the value `no`), the `access_info_monitor` command output is not in ASN.1 format. This argument is useful for the OpenVOS kernel OID mechanism interface with SNMP.



## Explanation

This `access_info_monitor` command enables you to display kernel variables or to change the value of kernel variables that are settable. (The command `netstat -64` displays many of these kernel variables with more usable formatting.)

The command output displays some or all of the following information, depending upon the arguments specified:

- **OID**—The SNMP object identifier (OID), which is predefined by a standards organization or by Stratus. Stratus OIDs begin with `1.3.6.1.4.1.458`.

An OID that represents a table index has the form `n.xN`, where `n.` represents one or more numbers and `N` represents one number (for example, `1.3.6.1.2.1.31.1.4.1.x1`).

The letter `v` indicates that the OID begins with `1.3.6.1.4.1.458.114.1.6`. For example, `1.3.6.1.2.1.31.1.1.1.v19` is a shortened form of `1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.1.31.1.1.1.19`.

- **title**—A mnemonic title (that is, variable name).
- **type**—`table`, `table entry`, `row`, `int32`, `int64`, `string`, or `binary`.

A type may be preceded by one or more of the following values, which provide information about the type:

U	unsigned
X	hexadecimal
4	IPv4 address
6	IPv6 address
M	MAC address

- **value**—The value of the variable.
- **SNMP syntax**—Additional information about the variable.

## Examples

This section provides examples of the `access_info_monitor` command and its output.

## Example 1

**access\_info\_monitor 1.3.6.1.2.1.1**

Walking the info\_monitor database for 1.3.6.1.2.1.1:

```
1.3.6.1.2.1.1.1  sysDescr string = 'OpenVOS Release 17.1.1ac' DisplayString (SIZE
(0..255))
1.3.6.1.2.1.1.2  sysObjectID string = '1.3.6.1.4.1.458.104.4.4' OBJECT IDENTIFIER
1.3.6.1.2.1.1.3  sysUpTime UX int32 = 0xF221E4C TimeTicks
1.3.6.1.2.1.1.4  sysContact string = '<your_system_administrator and email
contact>' DisplayString (SIZE (0..255))
1.3.6.1.2.1.1.5  sysName string = '<your_system_name>' DisplayString (SIZE
(0..255))
1.3.6.1.2.1.1.6  sysLocation string = '<your_system_location>' DisplayString (SIZE
(0..255))
1.3.6.1.2.1.1.7  sysServices int32 = 76 (application transport network)
sysServices
1.3.6.1.2.1.1.8  sysORLastChange UX int32 = 0x0 TimeStamp
1.3.6.1.2.1.1.9  sysORTable table sysOREntry
1.3.6.1.2.1.1.v3 vosSysHDUpTime UX int64 = 0xF221E4C TimeTicks64
1.3.6.1.2.1.1.v8 vosSysORHDLASTChange UX int64 = 0x0 TimeStamp64
```

## Example 2

**access\_info\_monitor 1.3.6.1.2.1.1.5 set '%s#m1'**

1.3.6.1.2.1.1.5 sysName string = '%s#m1'

## Example 3

**access\_info\_monitor 1.3.6.1.2.1.1 -exclude\_std\_oids**

Walking the info\_monitor database for 1.3.6.1.2.1.1:

```
1.3.6.1.2.1.1.9  sysORTable table sysOREntry
1.3.6.1.2.1.1.v3 vosSysHDUpTime UX int64 = 0xF221E4F TimeTicks64
1.3.6.1.2.1.1.v8 vosSysORHDLASTChange UX int64 = 0x0 TimeStamp64
```

## Example 4

**access\_info\_monitor 1.3.6.1.2.1.1 -no\_include\_titles**

Walking the info\_monitor database for 1.3.6.1.2.1.1:

```
1.3.6.1.2.1.1.1  - string = 'OpenVOS Release 17.1.1ac' DisplayString (SIZE
(0..255))
1.3.6.1.2.1.1.2  - string = '1.3.6.1.4.1.458.104.4.4' OBJECT IDENTIFIER
1.3.6.1.2.1.1.3  - UX int32 = 0xF221E51 TimeTicks
1.3.6.1.2.1.1.4  - string = '<your_system_administrator and email contact>'
DisplayString (SIZE (0..255))
1.3.6.1.2.1.1.5  - string = '%s#m1' DisplayString (SIZE (0..255))
```

*(Continued on next page)*

```
1.3.6.1.2.1.1.6   - string = 'Maynard' DisplayString (SIZE (0..255))
1.3.6.1.2.1.1.7   - int32 = 76 (application transport network) sysServices
1.3.6.1.2.1.1.8   - UX int32 = 0x0 TimeStamp
1.3.6.1.2.1.1.9   - table sysOREntry
1.3.6.1.2.1.1.v3  - UX int64 = 0xF221E51 TimeTicks64
1.3.6.1.2.1.1.v8  - UX int64 = 0x0 TimeStamp64
```

### Example 5

To comply with RFC-1112, OpenVOS, by default, does not send Internet Group Management Protocol (IGMP) reports to the all-routers multicast address of 224.0.0.2. To enable OpenVOS to send reports to 224.0.0.2, in addition to any other intended address, issue the following command:

```
access_info_monitor 1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.1.4.139 set 1
```

As recommended by RFC-1112 and other standards, OpenVOS, by default, retransmits IGMP router reports only once or twice when a host first joins a multicast group. To enable six retransmissions of IGMP router reports, issue the following command:

```
access_info_monitor 1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.1.4.140 set 6
```

### Related Information

See [“Problem 8: Communication Problems Associated with a Large Number of Sockets” on page 10-7](#) for information on using the `access_info_monitor` command in a troubleshooting situation.

See the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420) for information about using the `access_info_monitor` command to set values for the `tcpVosRecvPshFlagHandling` variable.

## arp

### Purpose

The `arp` command displays the logical-to-physical address translation table used by the protocol ARP. The ARP table converts a host's IP address to the appropriate physical 48-bit MAC address. General users can issue this command to simply display ARP table entries for the local subnet; only system administrators (privileged users) can issue this command to delete or set an entry in the ARP table.

#### NOTE \_\_\_\_\_

The `arp` command supports only IPv4 addresses.

### Display Form

```
----- arp -----
hostname: 
-all:    no
-network:
-delete:  no
-set:     no
mac_addr:
```

### Command-Line Form

```
arp [hostname]
    [-all]
    [-network string]
    [-delete]
    [-set]
    [mac_addr]
```

### Arguments

► *hostname*

Specifies the host name or IP address associated with a network interface on a given host. If you are using the `hosts` database file instead of the `resolv.conf`

database file with a DNS server, host names and IP addresses must reside in the `hosts` database file. To specify an IP address, use standard dot notation (for example, `172.16.2.14`).

#### NOTE

Always specify a complete IP address in *hostname* to avoid interpretation errors. See [route](#) for more information.

#### ► `-all`

CYCLE

Displays all of the current ARP entries in the table. Note that if multiple network interfaces are connected to the same subnet (each configured as a logical network interface), the command may display the ARP entry for a given host multiple times, once for each network interface.

Do not specify the `-all` argument if you specify an IP address; when you specify an IP address, only the entry associated with that IP address is displayed. By default, the command does not display all of the current ARP entries in the table.

#### ► `-network string`

Displays the current ARP entries for the network interfaces with the specified network number (the network number portion of the local IP address, such as `172.16.2`). This argument applies if your module provides multiple STCP connections to different subnets.

#### ► `-delete`

CYCLE

For privileged users, deletes an existing ARP table entry for the host specified with the *hostname* argument. By default, this command does not delete an entry from the ARP table.

#### ► `-set`

CYCLE

For privileged users, creates an ARP table entry that maps the specified host's IP address to a specified MAC address. By default, this command does not create an ARP table entry.

#### ► `mac_addr`

For privileged users creating an ARP table entry, the 48-bit MAC address for the specified host. Specify the MAC address as 6 hexadecimal bytes separated by either hyphens or colons (for example, `00-00-A6-00-EC-60` or `00:00:A6:00:EC:60`).

## Explanation

The `arp` command displays the contents of the ARP table and also enables system administrators to manipulate an entry in the ARP table. Entries in the ARP table are

either generated automatically by ARP response packets received from the LAN or entered manually with this command.

If an entry is generated by an ARP response packet, the entry is labeled `temp` in the `Type` field of the output. A `temp` entry is deleted after 10 minutes (the `Life` field identifies the remaining time in minutes). If another entry is needed, another ARP response packet generates another entry, which also has a lifetime of 10 minutes.

If an entry is generated manually with this command, it is considered permanent until the network is brought down. The `Type` field identifies the entry as `perm`.

You would generally manipulate the ARP table entries under the following conditions.

- when you are connecting a host that does not support ARP to the network and you therefore need to create an ARP table entry for that host
- when a host is disconnected from the network and you need to delete the permanent entry from the table

If you do not provide any information with the command, the command displays the correct usage.

The `netstat` command displays ARP statistics.

STCP continues to use ARP cache entries even after the entries expire; the entries have a grace period during which the system continues to use them while they are being replenished.

## Examples

This section provides examples of the `arp` command.

### Example 1

The following `arp` command displays all of the entries in the ARP table.

```
arp -all
```

Internet Address	MAC Address	Type	Life
172.16.2.9	00-00-0C-01-76-13	temp	10 mins
172.16.2.55	00-00-A5-70-1E-05	temp	10 mins

If no entries exist in the ARP table, the command returns the following message.

```
arp: no entries in table
```

In this case, issue the `ping` command to verify the existence of a host on your local subnet. If the `ping` command is successful, issuing the `arp` command again should display an entry for the host “pinged,” as in the following example:

**ping m28**

```
Pinging host m28.corp.com (m11) : 172.16.2.46
ICMP Echo Reply:TTL 60 time = 15 ms
ICMP Echo Reply:TTL 60 time = 5 ms
ICMP Echo Reply:TTL 60 time = 5 ms
ICMP Echo Reply:TTL 60 time = 5 ms
Host m28.corp.com replied to all 4 of the 4 pings
```

**arp m28**

Internet Address	MAC Address	Type	Life
172.16.2.46	00-00-A8-8A-86-CB	temp	10 mins

### Example 2

The following `arp` command sets a MAC address for a specified host. (The entry would subsequently appear as permanent in the ARP table.)

```
arp -set 172.16.2.50 00-00-A4-72-1F-03
```

The command verifies the addition by displaying the following line.

```
172.16.2.50 mapped to mac address 00-00-A4-72-1F-03
```

### Example 3

The following `arp` command deletes the ARP entry for the host with the IP address 172.16.2.50.

```
arp -delete 172.16.2.50
```

The command verifies the deletion by displaying the following line.

```
Mapping for 172.16.2.50 deleted
```

## Error Messages

The following table lists some of the error messages returned by the `arp` command.

Error Message	Description
<code>arp: only superuser can add arp entries</code>	You must be privileged to add an <code>arp</code> entry.
<code>arp: only superuser can delete arp entries</code>	You must be privileged to delete an <code>arp</code> entry.
<code>arp: invalid hostname or Internet address <i>hostname</i></code>	You specified an invalid host name or IP address. Check your <code>hosts</code> database file.
<code>arp: invalid network <i>network</i></code>	You specified an invalid network.
<code>arp: invalid mac address</code>	You specified an invalid MAC address.
<code>Unable to add entry for <i>hostname</i></code>	STCP could not add the entry you specified.

## Related Information

See [“The `hosts` File and Host Name Information” on page 5-7](#) for more information about the hosts available on the network. Also, see the command descriptions for the [netstat](#) and [ping](#) commands later in this chapter.



# dlmux\_admin

**Privileged**

## Purpose

The `dlmux_admin` command instructs SDLMUX (that is, the STREAMS Data-Link Multiplexer communications driver) to perform a specified action for a configured network interface.

## Display Form

```
----- dlmux_admin -----
device_name: ████████████████████████████████████████████████████████████
action:      get_status
-partner:
```

## Command-Line Form

```
dlmux_admin device_name
           action [-partner new_partner_name]
```

## Arguments

- ▶ *device\_name* **Required**  
 The symbolic OpenVOS device name of a configured network interface on the current module. You must precede the name with the pound sign (#). Specify the name of a port (as the name appears in the port's `devices.tin` file entry) of an Ethernet PCI adapter. Specify the name of an SDLMUX-group interface when you specify the value `sdlmux_status` for the *action* argument.
- ▶ *action* CYCLE **Required**  
 The action that SDLMUX performs on the network interface specified in the *device\_name* argument. [Table 9-2](#) lists the values for the *action* argument. For more information about the *action* arguments, see the [Explanation](#) and [Examples](#) sections later in this command description.

**Table 9-2. Values of the *action* Argument of the *dlmux\_admin* Command** (Page 1 of 2)

Action	Description
<code>add_partner</code>	<p>Adds a partner to an existing network interface. When you specify the action <code>add_partner</code>, you must specify the device name of the partner with the <code>-partner</code> argument. The <code>add_partner</code> action adds a partner to the port specified in the <code>device_name</code> argument. The existing port and the new partner must be configured in an existing SDLMUX-group interface; the existing SDLMUX-group interface, the existing port, and the new partner must all have correct <code>devices.tin</code> file entries.</p>
<code>delete_partner</code>	<p>Deletes the partner of the port specified in the <code>device_name</code> argument. Before you delete an active partner, you must make it inactive and then delete it by issuing the <code>dlmux_admin</code> command twice: first, to perform a manual hardware failover by using the action <code>failover</code>, then to delete the partner by using the action <code>delete_partner</code>. You cannot delete an active partner (that is, a partner that is currently handling communications I/O); attempting to delete an active partner generates an error.</p> <p>Do not specify the action <code>delete_partner</code> with the <code>-partner</code> argument.</p>
<code>failover</code>	<p>Performs a manual hardware failover. For the <code>device_name</code> argument, you must specify the name of a network interface currently handling I/O; the <code>failover</code> action forces I/O from <code>device_name</code> to its partner. The partner cannot be active; if you try to move I/O to an active partner, the command generates an error. Do not specify the <code>-partner</code> argument with the <code>failover</code> action.</p> <p>Note that the action <code>failover</code> is typically used for testing. Frequent use of <code>failover</code> results in lost packets and poor performance.</p>
<code>get_status</code> (the default)	<p>Displays the following statistics.</p> <ul style="list-style-type: none"> <li>• the SDLMUX-group interface name</li> <li>• the complete device name of the interface</li> <li>• the state of the interface (UP or DOWN)—The state is preceded by <code>ACTIVE</code> when this interface is actively handling communications I/O. The state may be <code>UP</code> (network connection lost) when a cable is disconnected.</li> <li>• either the complete device name of the interface's partner or <code>none</code> if no partner exists</li> <li>• either the state of the partner (UP or DOWN), which is preceded by <code>ACTIVE</code> when this partner is actively handling communications I/O, or <code>none</code> if no partner exists</li> </ul>

**Table 9-2. Values of the *action* Argument of the *dlmux\_admin* Command** (Page 2 of 2)

Action	Description
<code>init_sdlmux</code>	<p>Initializes the SDLMUX-group interface that contains the port specified in the <i>device_name</i> argument and its partner, if one exists, as configured in <i>devices.tin</i> entries; also loads SDLMUX if it is not yet loaded. The SDLMUX-group interface becomes the logical network interface for the port(s) in the group. After you initialize a port, you must issue the <code>ifconfig</code> command with the argument <code>-add</code> to define the SDLMUX-group interface to the STCP protocol stack. The status of the port specified in the <i>device_name</i> argument of the <i>dlmux_admin</i> command becomes <code>ACTIVE UP</code> and the status of its partner, if one exists, becomes <code>UP</code> (if the command completes with no errors).</p>
<code>sdlmux_status</code>	<p>Displays the following statistics for an SDLMUX-group interface.</p> <ul style="list-style-type: none"> <li>• the SDLMUX-group interface name</li> <li>• the device name of the port in the group</li> <li>• the state of the devices (<code>UP</code> or <code>DOWN</code>)—The state is preceded by <code>ACTIVE</code> when the interface is actively handling communications I/O. The state may be <code>UP</code> (network connection lost) when a cable is disconnected.</li> </ul> <p>When you specify <code>sdlmux_status</code> for the <i>action</i> argument, you must specify the name of an SDLMUX-group interface for the <i>device_name</i> argument.</p>
<code>uninit_sdlmux</code>	<p>Uninitializes the SDLMUX-group interface that contains the port specified in the <i>device_name</i> argument and its partner, if one exists, as configured in <i>devices.tin</i> entries. You <b>must</b> first, however, issue the command <code>ifconfig #SDLMUX_group_name -delete</code> to delete the SDLMUX-group interface. If <i>device_name</i> is the last active SDLMUX-associated network interface, the action <code>uninit_sdlmux</code> also unloads SDLMUX if it was loaded automatically.</p>

► `-partner new_partner_name`

The device name of a newly configured network interface that is to be the partner of the existing network interface specified in the *device\_name* argument. The *devices.tin* file must contain an entry for the new partner and an entry for the associated SDLMUX-group interface. When you specify the `-partner` argument, you must also specify the `add_partner` value of the *action* argument.

## Explanation

The *dlmux\_admin* command applies to network interfaces such as ports. You can use the following *dlmux\_admin* command to obtain the status of a port. The argument *device\_name* must be the name of one port in an SDLMUX-group interface.

```
dlmux_admin #device_name get_status
```

You can also use the `dlmux_admin` command with an SDLMUX-group interface name to obtain the status of the group. Specify an SDLMUX-group interface name for the `device_name` argument and `sdlmux_status` for the value of the `action` argument, as in the following command.

```
dlmux_admin #SDLMUX_group_name sdlmux_status
```

(You can also use the `dump_sdlmux` request of the `analyze_system` subsystem to obtain the device names of ports configured in an SDLMUX-group interface. For information about the `dump_sdlmux` request of the `analyze_system` subsystem, see the *OpenVOS System Analysis Manual* (R073).)

The `dlmux_admin` command also performs the following tasks.

- Initializes an SDLMUX-group interface—You can use the following `dlmux_admin` command to initialize an SDLMUX-group interface (that is, to initialize the port `device_name` and its partner if one exists as configured in the `devices.tin` file). The action `init_sdlmux` also loads SDLMUX if it is not yet loaded.

```
dlmux_admin #device_name init_sdlmux
```

The status of the port named in the `device_name` argument becomes `ACTIVE UP`; the status of its partner, if one exists, becomes `UP`.

You can initialize an SDLMUX-group interface even if one of the adapters in the group is broken or not present. (The missing or broken adapter must have a `devices.tin` file entry.) In such cases, the status of the partner that is broken or missing is `DOWN`, and the `dlmux_admin` command issues the message `WARNING: Device device_name hardware is broken or absent`. In addition, the message `SDLMUX: device name device_name is broken` is logged in the `syserr_log.date` file.

After you initialize the port(s), you must define the associated SDLMUX-group interface to the STCP protocol stack by using the `ifconfig` command with the argument `-add`.

For additional information about defining SDLMUX-group interfaces and initializing ports, see [Chapter 3](#).

- Uninitializes an SDLMUX-group interface—You can use the following `dlmux_admin` command to uninitialize an SDLMUX-group interface (that is, to uninitialize the port `device_name` and its partner, if one exists, as configured in the `devices.tin` file).

```
dlmux_admin #device_name uninit_sdlmux
```

If the port `device_name` is the last active SDLMUX-associated network interface, the command also unloads the SDLMUX driver if it was loaded automatically. Note

that before you issue this `dlmux_admin` command, you **must first** delete the associated SDLMUX-group interface by issuing the command `ifconfig #SDLMUX_group_name -delete` (for information about using the `ifconfig` command to delete an SDLMUX-group interface, see [“Deleting an SDLMUX-Group Interface and Its Ports” on page 3-27](#)).

- Deletes a partner—You can use the following `dlmux_admin` command to delete the partner of the port specified by the `device_name` argument, as long as the partner to be deleted is not handling communications I/O.

```
dlmux_admin #device_name delete_partner
```

Before you delete the partner, you must first make it inactive by performing a manual hardware failover.

- Adds a partner—You can use the following `dlmux_admin` command to add a partner to the port specified by the `device_name` argument.

```
dlmux_admin #device_name add_partner -partner partner_name
```

You **must** issue this command to partner a newly configured network interface with an existing SDLMUX-associated network interface if you want the change to take effect during the current bootload. The command enables you to make this configuration change immediately, while the existing network interface is actively handling communications I/O. Do not use this command when you are defining two **new** network interfaces as partners. For more information, see [Chapter 3](#).

You can add a partner to an SDLMUX-group interface even if one of the adapters in the group is broken or not present. (The partner must have a `devices.tin` file entry.) In such cases, the `dlmux_admin` command issues a warning.

If you replace one type of Ethernet adapter with a different type, you do **not** receive a warning that the adapter is of a different type, even if the network experiences problems.

- Requests a manual hardware failover—You can use the following `dlmux_admin` command to request that the partner of the specified SDLMUX-associated network interface begin handling communications I/O. The command will fail if you try to move communications I/O to an active partner.

```
dlmux_admin #device_name failover
```

#### NOTE

The action `failover` is typically used for testing. Frequent use of `failover` results in lost packets and poor performance.

The `dlmux_admin` command uses the network interface that you specify in the *device\_name* argument to determine how to perform the requested action. If, for example, you specify `enet.m1.13-1.1` as the device name of an existing SDLMUX-associated network interface, and you specify `init_sdlmux` for the *action* argument, the command assigns a partner to `enet.m1.13-1.1` as configured in its `devices.tin` file entry. If you later specify `enet.m1.13-1.1` as the device name of an active SDLMUX-associated network interface and then specify `failover` as the action, the command causes a manual failover from that interface to its partner.

## Examples

### NOTE

The output of the `dlmux_admin` command displays each device name with a system-name prefix. For example, the command output displays `%s1#enet.m1.13-1.1` for the device `enet.m1.13-1.1`.

The following `dlmux_admin` command initializes the SDLMUX-group interface that contains the device `enetA.m1.10-6.0` and its partner, as configured in the `devices.tin` file entries. The device `enetA.m1.10-6.0` is an embedded Ethernet PCI adapter, and its partner is the embedded adapter that has the name `enetA.m1.11-6.0`.

```
dlmux_admin #enetA.m1.10-6.0 init_sdlmux
```

If the preceding command is successful and if the SDLMUX-group interface has been defined (that is, the `ifconfig SDLMUX_group_name` command has been issued), you can issue the following `dlmux_admin` command to check the status of the device.

```
dlmux_admin #enetA.m1.10-6.0 get_status
```

```
Group Name:      #sdlmuxA.m1.10-6.0.11-6.0
Device Name:     %s1#enetA.m1.10-6.0
Adapter State:   ACTIVE UP
Partner:        %s1#enetA.m1.11-6.0
Partner State:   UP
```

You can also check the status of the device by using the SDLMUX-group interface name. The following `dlmux_admin` command checks the status of the SDLMUX-group interface `sdlmuxA.m1.10-6.0.11-6.0`.

```
dlmux_admin #sdlmuxA.m1.10-6.0.11-6.0 sdlmux_status
```

```
Group Name:          #sdlmuxA.m1.10-6.0.11-6.0
Device Name:         %s1#enetA.m1.10-6.0
Adapter State:       ACTIVE UP
Partner:             %s1#enetA.m1.11-6.0
Partner State:       UP
```

The following `dlmux_admin` command requests a manual hardware failover to force Ethernet I/O from the active network interface `enetA.m1.10-6.0` to its partner `enetA.m1.11-6.0`.

```
dlmux_admin #enetA.m1.10-6.0 failover
```

After you issue the `dlmux_admin` command to cause the failover, issue the following `dlmux_admin` command to verify the status of the partner.

```
dlmux_admin #enetA.m1.11-6.0 get_status
```

```
Group Name:          #sdlmuxA.m1.10-6.0.11-6.0
Device Name:         %s1#enetA.m1.11-6.0
Adapter State:       UP
Partner:             %s1#enetA.m1.10-6.0
Partner State:       ACTIVE UP
```

The following `dlmux_admin` command uninitializes the SDLMUX-group interface that contains both the embedded port `enetA.m1.11-6.0` and its partner (as defined in the `devices.tin` file). Note, however, that before you can delete a partnered port, you **must first** delete the associated SDLMUX-group interface by issuing the command `ifconfig #SDLMUX_group_name -delete`.

```
dlmux_admin #enetA.m1.11-6.0 uninit_sdlmux
```

If you issue the `dlmux_admin` command again to verify the status, you will receive the following error messages.

```
sdlmux_admin: Device name %s1#enetA.m1.11-6.0 is not
+ initialized yet
dlmux_admin: The specified device name is not known to the
+ system.
```

## Error and Status Messages

The `dlmux_admin` command returns error and status messages to the system console if the information you supply is incorrect or cannot be processed as specified.

[Table 9-3](#) lists and describes common error and status messages of the `dlmux_admin` command.

**Table 9-3. Common Error and Status Messages of the `dlmux_admin` Command** (Page 1 of 2)

Message	Description
<code>dlmux_admin: An invalid argument value was given.</code>	You have specified an invalid argument value. For example, you may have specified an invalid device name.
<code>dlmux_admin: Cannot open.</code>	The specified device cannot be opened.
<code>dlmux_admin: delete device_name from SDLMUX_group_name</code>	The specified SDLMUX-associated device was deleted from <code>SDLMUX_group_name</code> .
Device <code>device_name</code> does not specify a valid PCI slot number ( <code>number</code> ).	The value ( <code>number</code> ) of the <code>hardware_address</code> field of the <code>devices.tin</code> file entry for <code>device_name</code> is incorrect.
<code>dlmux_admin: Device device_name hardware is broken or absent.</code>	The specified device is present but in an unusable state (for example, the adapter is broken), or the adapter is entirely absent from its slot.
<code>dlmux_admin: Device name device_name is not initialized yet.</code>	The SDLMUX-group interface with which this device is associated has not yet been initialized. (You initialize the SDLMUX-group interface using the command <code>dlmux_admin #device_name init_sdlnmux.</code> )
<code>dlmux_admin: Device device_name hardware is not recognized.</code>	The device is present, but it is not of a recognizable type. For example, the device is not a gigabit Ethernet PCI adapter.
<code>dlmux_admin: Device device_name specifies an invalid board number (number).</code>	The value ( <code>number</code> ) of the <code>hardware_address</code> field of the <code>devices.tin</code> file entry for <code>device_name</code> is incorrect.
<code>dlmux_admin: Devices device_name and second_device_name reside in the same I/O board (number).</code>	The two PCI adapters in an SDLMUX-group interface are located in the same CPU-I/O enclosure.
<code>dlmux_admin: Invalid controller slot number.</code>	You may have tried to add a partner whose hardware type does not match that of the existing SDLMUX-associated network interface.



**Table 9-3. Common Error and Status Messages of the dlmux\_admin Command** (Page 2 of 2)

Message	Description
dlmux_admin: Invalid device name.	The specified device does not exist.
dlmux_admin: Invalid -duplex between partner and device.	Values for the -duplex argument in the parameters field are not equal in the devices.tin file entries for two devices in an SDLMUX-group interface.
dlmux_admin: Invalid operation for this device type.	You may have specified the action add_partner or delete_partner for a network interface.
dlmux_admin: Invalid -speed between partner and device.	Values for the -speed argument in the parameters field are not equal in the devices.tin file entries for two devices in an SDLMUX-group interface.
dlmux_admin: Mismatching -sdlmux parameter fields for device_name and second_device_name.	Values for the -sdlmux argument in the parameters field are not equal in the devices.tin file entries for two devices in an SDLMUX-group interface.
dlmux_admin: No sdlmux group is specified for this device.	You have specified a valid device name for a port, but the devices.tin entry for this port does not specify an SDLMUX-group interface (that is, the entry does not contain the -sdlmux argument in the parameters field).
dlmux_admin: Specified device already in use.	The specified device is already in use.
dlmux_admin: The partner of device device_name is not second_device_name.	The devices.tin file entries for two devices in an SDLMUX-group interface do not provide consistent information.
dlmux_admin: The requested operation is inconsistent with the current device state.	You have tried to perform an action that is invalid given the current state of the specified network interface. For example, the command generates this error if you try to add a partner to a network interface that already has a partner, or if you try to delete the active partner of a partnered network interface.
dlmux_admin: The specified device name is not known to the system.	SDLMUX could not find the device name that you specified for the network interface (or its partner). The device may not have an entry in the devices.table file or the device may not have been initialized.
dlmux_admin: Wrong device type for this operation.	You may have specified the name of a device that is not a port of an Ethernet PCI adapter.

## Related Information

See the following documentation for related information.

- For information about the network interfaces supported by STCP on ftServer systems running OpenVOS, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).
- For information about creating device entries for the different types of network interfaces, see [Chapter 3](#).

# ftp

## Purpose

The `ftp` command starts an FTP session and enables you to transfer one or more files to and from a remote host and perform related operations.

## Display Form

```
----- ftp -----
host:      █
port:      21
-debug:    no
-verbose:  yes
-globbing: yes
-interactive: yes
-autologin: yes
-os_break: no
-passive:  no
```

## Command-Line Form

```
ftp [host]
    [port]
    [-debug]
    [-no_verbose]
    [-no_globbing]
    [-no_interactive]
    [-no_autologin]
    [-os_break]
    [-passive]
```

## Arguments

### ► *host*

### Required

Specifies the remote host with which you want to establish a connection for the FTP session. Specify either a host name or an IPv4 or IPv6 address. (Host names and IP addresses are defined in the `hosts` database file or by a DNS server. IP

addresses are defined using dot notation.) If you do not specify a host, you begin an FTP session in FTP subcommand mode (you see the prompt `ftp>`), without opening a connection to a remote host. In FTP subcommand mode, you can issue the `open` subcommand to request a connection to a remote host.

► `port`

Specifies the port number on the remote host for the FTP session. The range of values is 1 to 65535; the default value is 21, which is the standard `ftp` command port.

► `-debug`

CYCLE

Enables debugging mode. When debugging mode is enabled, any subcommand that you issue causes FTP to display the actual FTP protocol command that it sends to the remote host, preceded by the string `--->`. By default, debugging mode is disabled.

► `-no_verbose`

CYCLE

Disables verbose mode. When verbose mode is enabled (the default), FTP displays responses from the server and file-transfer statistics. Within an FTP session, you can change the setting of this mode with the `verbose` FTP subcommand.

► `-no_globbing`

CYCLE

Disables globbing (global file-name expansion mode) and causes FTP to treat every file name as a literal. When globbing mode is on (the default), you can use star names and abbreviations, and FTP expands each name and processes all names that match the star name or abbreviation. You can also control the setting for this mode by issuing the `glob` FTP subcommand from FTP subcommand level.

► `-no_interactive`

CYCLE

Disables interactive-prompting mode and causes FTP to perform certain file transfers or file deletion without prompting for verification. Interactive-prompting mode is enabled by default to ensure that you verify actions that apply to multiple files or files that could be overwritten. You can also control the setting for this mode by issuing the `prompt` FTP subcommand from FTP subcommand level.

► `-no_autologin`

CYCLE

Disables auto-login mode and prevents FTP from checking your `.netrc` file, which provides the information required to perform an automatic login to the remote host. By default, auto-login mode is enabled, which enables FTP to check your `.netrc` file to handle user ID/password verification for the remote connection. If it can find the file, it will automatically log you in. (For information about the structure of the `.netrc` file, see the *OpenVOS STREAMS TCP/IP User's Guide* (R421).)

► `-os_break`

CYCLE

Turns on default break-handling mode. When this mode is on, FTP uses the default OpenVOS break handler; when this mode is off, FTP uses its own break handlers

during data transfer. The FTP break handlers enable you to abort the current data transfer without ending the FTP process (for example, when you issue your terminal's break command during a transfer, FTP aborts the transfer but returns you to FTP subcommand level). When default break handling is on, if you issue a break command and stop the process, OpenVOS ends your FTP session and returns you to OpenVOS command level. Stratus recommends that you leave this mode off (it is off by default).

► `-passive`

CYCLE

Turns on passive mode. In active mode FTP (the default), the client opens the command channel and the server opens the data channel. In passive mode FTP, the client opens the command and data channels.

## Explanation

The `ftp` command invokes the FTP client and starts an FTP session. You can start an FTP session by trying to connect to a remote host, or you can enter FTP subcommand mode and then issue FTP subcommands. While you are connected to a remote host, you can perform file-transfer operations. By default, the file-transfer type is ASCII, but you can use FTP subcommands to change the file type (for example, to binary or image, sequential for Stratus-to-Stratus transfers, or Tenex).

If you want to use the `ftp` command to transfer a non-streams file or you want to maintain some or all of the file's attributes, you first need to bundle the file by issuing the `bundle` command, specifying the `-output_format` argument with the value `for_ftp`.

If you use the `ftp` command in binary mode to transfer a 64-bit stream file to a module that does not support 64-bit stream files, the file becomes a normal stream file.

If you use the `ftp` command to transfer a normal stream file to a module that supports 64-bit stream files, the file becomes a 64-bit DAE-8 (by default) stream file. If you use `ftp` in binary mode and the `get` subcommand to transfer a normal stream file to a module that supports 64-bit stream files, `ftp` creates a 64-bit stream file; therefore, transfer is limited to 16 GB. You can use the `append` subcommand to create targets of `put` subcommands beforehand, to allow for up to 550 GB if necessary, when the target is on an OpenVOS module. You can use the `lappend` subcommand to set characteristics of the target of the `get` subcommand. (For complete information on 64-bit stream files, see the `create_file` command description in the *OpenVOS Commands Reference Manual* (R098).)

[Table 9-4](#) lists the STCP FTP (client) subcommands in alphabetical order and provides brief descriptions of the subcommands. For more information about the subcommands, see the *OpenVOS STREAMS TCP/IP User's Guide* (R421).

**Table 9-4. FTP Subcommands** (Page 1 of 5)

Subcommand	Description
! <i>[command_line]</i>	Creates a subprocess from which you can issue one or more OpenVOS commands.
\$ <i>macro_name</i> <i>[argument...]</i>	Executes the specified macro.
.. <i>command_line</i>	Issues an VOS command line and returns to the FTP session.
? <i>[subcommand]</i> or help	Provides information about the supported subcommands.
append <i>local_file</i> <i>[remote_file]</i>	Transfers a file from the local host to the remote host and appends the file to the end of an existing file.
ascii	Sets the file-transfer type to ASCII.
bell	Changes the setting for bell mode.
binary <i>[record_size]</i>	Sets the file-transfer type to binary.
bye	Ends the FTP session, returning your process to command level.
case	Changes the setting for case-mapping mode.
cd <i>remote_directory</i>	Changes the current directory on the remote host.
cdup	Changes the current directory on the remote host to the parent of the current directory.
close	Closes the connection with the remote host but maintains the FTP session.
cr	The default setting is permanently on. CR-stripping mode strips out ASCII CR characters from an ASCII file when it is received.
debug	Changes the setting for debugging mode.
delete <i>remote_file</i>	Deletes the specified file on the remote host.
dir <i>[remote_directory [local_file]]</i>	Lists the contents of a directory on the remote host, with details.
disconnect	Closes the connection with the remote host but maintains the FTP session.

**Table 9-4. FTP Subcommands** (Page 2 of 5)

Subcommand	Description
<code>form</code>	Displays the default file-transfer format.
<code>get remote_file [local_file]</code>	Transfers a file from the remote host to the local host.
<code>glob</code>	Changes the setting for global file-name expansion mode.
<code>hash</code>	Changes the setting for hash-mark mode.
<code>image [record_size]</code>	Sets the file-transfer type to <code>image</code> . This command performs the same function as the <code>binary</code> subcommand.
<code>lappend remote_file [local_file]</code>	Transfers a file from the remote host to the local host and appends the file to the end of an existing file.
<code>lcd [local_directory]</code>	Changes the current directory on the local host.
<code>literal arg1 [arg2...]</code>	Sends one or more verbatim strings to the FTP server on the remote host.
<code>ls [remote_directory [local_file]]</code>	Lists the contents of a directory on the remote host, with details.
<code>macdef macro_name</code>	Defines a macro.
<code>mdelete remote_file...</code>	Deletes multiple files on the remote host. Separate each file name with a space or use a star name.
<code>mdir remote_directory... local_file</code>	Lists the contents of multiple directories on the remote host, with details. Separate each directory name with a space or use a star name.
<code>mget remote_file...</code>	Transfers multiple files from the remote host to the current directory on the local host.
<code>mkdir remote_directory</code>	Creates a directory on the remote host.
<code>mls remote_directory... local_file</code>	Lists the contents of multiple directories on the remote host, without details.
<code>mode</code>	Displays the file-transfer transmission mode, which is stream.

**Table 9-4. FTP Subcommands** (Page 3 of 5)

Subcommand	Description
<code>modtime file_name</code>	Indicates the time at which a file on the remote host was last modified.
<code>mput local_file...</code>	Transfers multiple files from the local host to the current directory on the remote host.
<code>nlist [remote_directory [local_file]]</code>	Lists the contents of a directory on the remote host, without details. If you want the contents written to a file on the local host, specify a file name.
<code>nmap [inpattern outpattern]</code>	Changes the setting for file-name mapping mode. The <code>inpattern</code> argument specifies the mapping of the components in the source file name; <code>outpattern</code> specifies the mapping of the corresponding components in the destination file name.
<code>ntrans [inchars [outchars]]</code>	Changes the setting for character-translation mode. The <code>inchars</code> argument specifies the characters in the source file name that you want to translate; <code>outchars</code> specifies the corresponding translation characters in the destination file name.
<code>open host [port_number]</code>	Tries to open a connection with a remote host.
<code>passive</code>	Turns on passive mode. In active mode FTP (the default), the client opens the command channel and the server opens the data channel. In passive mode FTP, the client opens the command and data channels. If you are in passive mode, enter the <code>passive</code> subcommand again to switch back into active mode.
<code>prompt</code>	Changes the setting for interactive-prompting mode (which prompts you when you transfer or delete multiple files or could overwrite an existing file).
<code>proxy ftp_subcommand</code>	Allows you to issue subcommands relevant to the proxy connection.
<code>put local_file [remote_file]</code>	Transfers a file from the local host to the remote host.



**Table 9-4. FTP Subcommands** (Page 4 of 5)

Subcommand	Description
<code>pwd</code>	Displays (prints) the name of the current working directory on the remote host.
<code>quit</code>	Ends the FTP session and returns you to command level.
<code>quote arg_1 [arg_2...]</code>	Sends one or more verbatim strings to the FTP server on the remote host.
<code>recv remote_file [local_file]</code>	Transfers a file from the remote host to the local host. This subcommand performs the same function as the <code>get</code> subcommand.
<code>remotehelp [command]</code>	Displays information about the FTP protocol subcommands that the remote host supports.
<code>rename from to</code>	Renames a file on the remote host.
<code>reset</code>	Not implemented in STCP FTP.
<code>rhelpt command</code>	Displays information about the FTP protocol subcommands that the remote host supports.
<code>rmdir remote_directory</code>	Deletes a directory on the remote host.
<code>rstatus</code>	Displays the status of the current connection with the remote host.
<code>runique</code>	Changes the setting for receive-unique mode, which, when enabled, ensures that each file transferred to the local host has a unique name. By default, it is off.
<code>send local_file [remote_file]</code>	Transfers a file from the local host to the remote host. This subcommand performs the same function as the <code>put</code> subcommand.
<code>sendport</code>	Changes the setting for sendport mode, which is on by default, enabling FTP to send the FTP protocol command <code>PORT</code> to the server on the remote host when establishing a connection for file transfer.
<code>sequential</code>	Sets the file-transfer type to sequential, which allows OpenVOS sequential files to be transferred between Stratus modules.
<code>size file_name</code>	Indicates the size, in bytes, of a file on the remote host.

**Table 9-4. FTP Subcommands** (Page 5 of 5)

Subcommand	Description
status	Displays status information for the current FTP session.
struct	Displays the default file-transfer structure, which is <code>file</code> . The <code>file</code> mode specifies that a file is sent as a stream of bytes with no recognition of internal structure such as records or pages.
sunique	Changes the setting for store-unique mode, which, when enabled, ensures that each file that is transferred to the remote host has a unique name. By default, it is off.
system	Indicates the system type of the remote host.
tenex	Sets the file-transfer type to Tenex, which is suitable for transferring a file to or from a Tenex machine.
type [ <i>type_name</i> ] [ <i>record_size</i> ]	Sets the file-transfer type to <code>ascii</code> , <code>binary</code> , <code>image</code> , <code>seq</code> , or <code>tenex</code> . The <i>record_size</i> argument specifies a record length for the destination file when the type is <code>binary</code> or <code>image</code> .
user <i>user_name</i> [ <i>password</i> ]	Sends login data to the remote host, such as your user name and password.
verbose	Changes the setting for verbose mode, which, when enabled (the default), causes FTP to display responses from the remote host when you issue a subcommand.

## Examples

The following example shows an excerpt from a sample FTP session.

```
ftp m3
Connected to m3.corp.com.
220 sysm3 FTP server (FTP 1.0 for Stratus STCP) ready. (Compatible with
+OS TCP/IP)
User (m3.corp.com:Jane_Doe): jane
331 Password required for jane.
Password:
230 User Jane_Doe.SysAdmin logged in.
ftp> ?
```

(Continued on next page)

Commands may be abbreviated. Commands are:

!	close	ls	open	runique
..	cr	macdef	prompt	send
?	delete	mdelete	proxy	sendport
\$	debug	mdir	put	size
append	dir	mget	pwd	sequential
ascii	disconnect	mkdir	quit	status
bell	form	mls	quote	struct
binary	get	mode	recv	sunique
image	glob	modtime	remotehelp	system
bye	hash	mput	rename	tenex
case	help	nlist	rhel	type
cd	lcd	nmap	rmdir	user
cdup	literal	ntrans	rstatus	verbose

ftp> **cr**

Carriage Return stripping is ON by default and cannot be turned off.

ftp> **pwd**

257 "%sys#m3\_mas>SysAdmin>forms" is current directory.

ftp> **ls**

Files: 36 Blocks: 1675

w	2	seq	96-05-24 10:41:02	abbreviations
w	8	seq	96-06-17 15:27:02	as_file
w	4	fix-1	96-09-09 15:09:48	form.pm

ftp> **type binary**

200 Type set to I.

ftp> **status**

Connected to m3.corp.com.

No proxy connection.

Type: binary; Form: non-print; Structure: file

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

Store unique: off; Receive unique: off

Case: off, CR stripping: on

Ntrans: off

Nmap: off

Debugging: off; Hash mark printing: off

Use of PORT cmds: on

ftp> **cd <more\_forms**

250 CWD command successful.

ftp> **pwd**

257 "%sys#m3\_mas>SysAdmin>more\_forms" is current directory.

ftp> **get tm.fm**

250 PORT command successful.

150 Opening data connection for tm.frm (172.16.100.2,1227) (13566 bytes).

226 Transfer complete.

13566 bytes received in 0.29 seconds (46.14 Kbytes/sec)

ftp> **close**

221 Goodbye.

ftp> **quit**

ready 15:22:48

## Error Messages

The following table lists some of the common error messages associated with the `ftp` command.

Error Message	Description
<code>ftp: connect: Connection refused.</code> <code>ftp: no control connection for command</code>	The <code>ftpd</code> daemon process is not running.
<code>ftp: connect: Connection timed out.</code>	The remote STCP stack is down, or the host is down.
<code>ftp: connect: Network is unreachable.</code>	You specified an invalid IP address, or the network cannot be reached.
<code>?Invalid command</code>	You entered an invalid string as a command.
<code>host: Unknown host.</code>	You specified a host that is unknown to the system.

## Related Information

See “[The `hosts.allow` and `hosts.deny` Files and TCP Wrappers](#)” on page 5-10 for information about TCP wrappers functionality.

See the *OpenVOS Commands Reference Manual* (R098) for descriptions of the `bundle` command and the `display_file_status` command, which includes a list of file attributes.

See the *OpenVOS STREAMS TCP/IP User's Guide* (R421) for more information about using FTP.

# hostname

## Purpose

The `hostname` command enables you to set or display the name of the current host system.

## Display Form

```
----- hostname -----  
hostname: █
```

## Command-Line Form

```
hostname [hostname]
```

## Arguments

► *hostname*

The host name you want to assign to the current host. You must be logged in as a privileged user to set the host name of the module.

## Explanation

If you do not specify a host name, the command displays the current host name for the module if one has been set. Any user can issue the command from command level to display the host name. However, only a privileged user can set the host name for the module.

When you issue the `hostname` command to set the host name of the current host, the command generates a file called `host` in the `(master_disk)>system>stcp` directory. (Make sure that this file does not get deleted.) To set the host name for the module at each bootload, issue the `hostname` command from the `module_start_up.cm` file.

## Examples

The following example sets the host name for module %sys1#m1.

```
hostname m1
```

## Related Information

For information about STCP and the `module_start_up.cm` file, see [“Starting STCP on the Module” on page 6-2](#).

# ifconfig

**Privileged**

## Purpose

The `ifconfig` command enables you to add, delete, check, or change the status of a network interface for STCP on the module. You issue this command from the `start_stcp.cm` command macro to add each STCP logical network interface automatically at each bootload (the command macro is invoked from the `module_start_up.cm` file). You also issue this command from command level to add a new interface or IP address, or delete, verify, or change the status of a configured network interface.

## Display Form

```
----- ifconfig -----
interface:
name_or_address:
-state:
-netmask:
-description:
-scope_zones:
-ipv4:          yes
-add:           no
-forwb:         no
-mtu:
-all:          no
-alias:         no
-lifetimes:     no
-anycast:       no
-deprecated:    no
-prefix_lifetime:
-accept_router_adv: unchanged
-ip6_disabled:  unchanged
-prefer_source: unchanged
-euid64:        no
-ipv6:          yes
-delete:        no
-kalive:        yes
-broadcast_type:
-ask:           yes
-quiet:         no
-default_ip6_interface: unchanged
-tentative:     no
-auto_configure: no
-valid_lifetime:
-add_default_routers: unchanged
-neighbor_detect: unchanged
-auto_linklocal: unchanged
```

## Command-Line Form

```
ifconfig [interface]
         [name_or_address]
         [-state string]
         [-netmask string]
         [-description string]
         [-scope_zones integer(s)]
         [-no_ipv4]
         [-no_ipv6]
         [-add]
         [-delete]
         [-forwb]
         [-no_kalive]
         [-mtu number]
         [-broadcast_type number]
         [-all]
         [-no_ask]
         [-alias]
         [-quiet]
         [-lifetimes]
         [-default_ip6_interface value]
         [-anycast]
         [-tentative]
         [-deprecated]
         [-auto_configure]
         [-prefix_lifetime number]
         [-valid_lifetime number]
         [-accept_router_adv value]
         [-add_default_routers value]
         [-ip6_disabled value]
         [-neighbor_detect value]
         [-prefer_source value]
         [-auto_linklocal value]
         [-euid64]
```



## Arguments

### ► *interface*

The symbolic OpenVOS device name of a network interface on your module. You specify a device name when you create a device entry for a network interface in the `devices.tin` file, which resides in the directory `(master_disk)>system>configuration`. You **must** precede the device name with the number sign (#) when specifying the network interface. If you specify only the device name of the interface, the command displays the status of the network interface, which includes the settings of the flags, the IP address, the net mask, and the broadcast address.

### ► *name\_or\_address*

The name or IP address of the network interface. If you specify a name, it must be defined in the file `(master_disk)>system>stcp>hosts`, or a connection to a DNS server must exist through another interface. You can specify an IP address of type IPv4 or IPv6.

### NOTE

Stratus recommends that you specify an IP address in *name\_or\_address* rather than relying on the DNS. If you are adding the interface that will be used to contact the DNS, you will not be able to contact the DNS if you are relying on the DNS to provide the *name\_or\_address* value.

You can use classless inter-domain routing (CIDR) notation by specifying */N* after the name, where *N* is referred to as the prefix length and indicates the number of high-order 1 bits in the netmask. The default prefix length for IPv6 is 64. The default prefix length for IPv4 remains unfixed. See the [Explanation](#) for more information about prefixes.

If you include */N* when specifying *name* (of the *name\_or\_address* argument), you cannot specify the `-netmask string` argument.

### ► *-state string*

CYCLE

Enables or disables data transfer through the network interface specified by the *interface* argument. For *string*, specify either `up` or `down`; `up` enables data transfer through a network interface configured for STCP; `down` disables data transfer through the network interface. Specify `down` to temporarily halt data transfer through a network interface. After you mark an interface as `down`, you will be unable to perform the `ping` command with the interface's corresponding IP address. Specify `up` to resume data transfer on an interface that you have previously marked as `down` but that is still configured for STCP. If you delete an interface instead of just marking the state as `down`, you will not be able to bring the state of that interface up; instead, you must add the interface again. Remember

that you do not need to mark the state as `up` when you are adding an interface with the `-add` argument, or mark the state as `down` when you delete an interface.

► `-netmask string`

Enables the configuration of logical subnetworks (subnets) using the IP network number, specified as an IPv4 address. Specify the correct subnet mask if you are configuring subnets. Specify the mask in hexadecimal format (or dot notation). For more information about configuring subnets and a subnet mask, see “[Specifying a Network Mask to Identify Subnets](#)” on page A-13. If you issue the command from command level and specify the subnet mask for a network interface you are adding permanently, the `ifconfig` entry you create for the interface in the `start_stcp.cm` command macro must also specify a subnet mask. If you do not specify a subnet mask, the command generates a class mask (that is, it determines the mask from the network class of the IP address).

Note that you must specify IPv6 prefixes using CIDR notation.

If you included `/N` when specifying `name` (of the `name_or_address` argument), you cannot specify the `-netmask string` argument.

► `-description string`

Sets a description (`string`) for the interface. Once a description is set, you cannot issue the `ifconfig` command to reset `string` to a null value, but you can reset `string` to a single space.

► `-scope_zones integer(s)`

Specifies the scope zone IDs for multicast addresses on the interface. You can specify up to ten integers. The first integer corresponds to scope 4, the second integer corresponds to scope 5, with similar correspondences up to the tenth integer, which corresponds to scope 13, as in the table below:

(Page 1 of 2)

Scope Number	Scope
4	<i>Admin-Local</i>
5	<i>Site-Local</i>
6	<i>unassigned</i>
7	<i>unassigned</i>
8	<i>Organization-Local</i>
9	<i>unassigned</i>
10	<i>unassigned</i>

(Page 2 of 2)

11	<i>unassigned</i>
12	<i>unassigned</i>
13	<i>unassigned</i>

Use zero (0) to designate an unspecified scope zone ID.

You cannot set the following scope zone IDs:

- *Interface-Local* and *Link-Local* are the interface index.
- *Global* is always zero.

► `-no_ipv4`

CYCLE

Specifies whether the *name\_or\_address* argument can be an IPv4 address. The value *yes* (the default) enables the interface to use an IPv4 address. The value *no* specifies that *address* must not be an IPv4 address. In this case, *address* can be the unspecified IPv6 address (: :).

This argument does not control whether IPv4 is enabled on the interface because IPv4 is always enabled; however, the argument enables you to configure an interface with no IPv4 addresses defined on it.

► `-no_ipv6`

CYCLE

Specifies whether the *name\_or\_address* argument can be an IPv6 address. The value *yes* (the default) configures the interface to support IPv6 addresses and enables the interface to use an IPv6 address. The value *no* specifies that *address* must not be an IPv6 address.

Note that you can configure the interface to support IPv6 without specifying an IPv6 address for it.

► `-add`

CYCLE

Adds the network interface specified by the *interface* argument as an available interface for STCP (defines and brings the interface up). For more information, see the [Explanation](#). (If you are permanently adding the interface, edit the `start_stcp.cm` command macro to define the interface.) Unless you specify the `-add` argument, the command does not add a network interface.

► `-delete`

CYCLE

Deletes the network interface specified by the *interface* argument from the list of interfaces available to STCP (that is, it deletes the interface from the list and stops data transfer). When you delete an interface with the `-delete` argument, all routes with a gateway address that uses that interface are automatically deleted.

To change the IP address of a configured and operational network interface, delete the interface and add it back in with the new IP address. By default, the command prompts you to confirm that you want to delete the specified network interface. If you do not want the command to prompt you for verification, specify the `-no_ask` argument in addition to the `-delete` argument. For more information, see the [Explanation](#). (If you are permanently deleting an interface that has an entry in the `start_stcp.cm` command macro, remove the entry from the macro.) Unless you specify the `-delete` argument, the command does not delete a network interface.

► `-forwb`

CYCLE

Sets the forward-broadcast flag, which forwards network and subnet broadcast packets from other networks to the network associated with the specified network (the interface specified by the `interface` argument). By default, this command does not forward broadcast packets. A broadcast packet received from another network will only be transmitted to the specified network if you specify this argument. If you issue the command from command level and specify the `-forwb` argument for a network interface you are adding permanently, the `ifconfig` entry you create for the interface in the `start_stcp.cm` command macro must specify the `-forwb` argument. If you want the module to act as a router/gateway (that is, if the IP forwarding feature is enabled, as viewed or set with the [IP\\_forwarding](#) command), you may want the network interface to forward broadcasts. By default, the IP forwarding feature is enabled.

► `-no_kalive`

CYCLE

If you specify this argument, the keepalive flag is disabled for the specified interface. If you do not specify this argument, the keepalive flag is enabled for the specified interface.

The keepalive flag controls, in part, the keepalive function. To enable the keepalive function, the application **must** also set the `SO_KEEPALIVE` socket option. (For more information about setting the `SO_KEEPALIVE` socket option, see the description of the `setsockopt` function in the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420)). If the keepalive flag is disabled for an interface, the keepalive function will **not** be enabled for any connection using that interface regardless of whether the application set the `SO_KEEPALIVE` socket option or not.

If you issue the command from command level and specify the `-no_kalive` argument for a network interface you are adding permanently, the `ifconfig` entry you create for the interface in the `start_stcp.cm` command macro must also specify the `-no_kalive` flag argument.

► `-mtu number`

Specifies an unsigned integer to override the hardware's maximum transmission unit (MTU) or packet size. The default value is 1500. The value actually used is the minimum of the value specified by the `-mtu` argument and the MTU supported by

the underlying driver. For information on MTU and Ethernet PCI adapters, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

If you issue the command from command level and specify the `-mtu` argument for a network interface that you are adding permanently, the `ifconfig` entry you create for the interface in the `start_stcp.cm` command macro must also specify the MTU value.

► `-broadcast_type number`

Specify the value 0 for the broadcast type to have applications that broadcast on this network interface use a 0s type of broadcast; specify the value 1 for a 1s type of broadcast (the default). If you issue the command from command level and specify a broadcast type for a network interface you are adding permanently, the `ifconfig` entry you create for the interface in the `start_stcp.cm` command macro must specify the correct broadcast type.

See “[STCP Options](#)” on page A-21 for a description of terms such as broadcast type, keepalive, and subnet mask.

► `-all`

CYCLE

The `-all` argument is retained for compatibility with older versions of the `ifconfig` command. The argument provides the same functionality as issuing the `ifconfig` command with no arguments.

► `-no_ask`

CYCLE

Suppresses the display of a prompt asking you to verify that you want to delete the specified network interface or alias IP address. By default, the command prompts you to verify that you want to delete the specified network interface or alias IP address (see Example 6 later in this command description for an example with the prompt).

► `-alias`

CYCLE

Adds an additional IP address to a network interface. You can specify only one alias IP address per each `ifconfig` command. A network interface can have any number of alias IP addresses. By default, the command does not add an additional IP address to the same interface.

► `-quiet`

CYCLE

Specifies the amount of information the command displays when you add an interface using the `-add` or `-alias` argument. To suppress information, specify the value `yes`. By default (the value `no`), the command displays all information.

► `-lifetimes`

CYCLE

Specifies whether the command displays the lifetime of IPv6 addresses. To display IPv6 lifetimes, specify the value `yes`. By default (the value `no`), the command suppresses the display of IPv6 lifetimes.

▶ `-default_ip6_interface value`

CYCLE

Specifies whether the interface is the default IPv6 interface. Values are:

- `yes`—Sets the interface as the default IPv6 interface.
- `no`—Ensures that this interface is not the default.
- `unchanged` (the default)—Setting remains unchanged.

▶ `-anycast`

CYCLE

Specifies whether the *name\_or\_address* argument is an anycast address, when *address* is of type IPv6. To configure *address* as an anycast address, specify the value `yes`. By default (the value `no`), *address* is not an anycast address. This argument is relevant only when adding an IPv6 address.

▶ `-tentative`

CYCLE

Specifies whether the state of the *name\_or\_address* argument is tentative when the interface is down and *address* is of type IPv6. By default (the value `no`), *name\_or\_address* is not tentative when the interface is down. This argument is relevant only when adding an IPv6 address. You typically do not need to specify a value for this argument.

The state of the *name\_or\_address* argument is always tentative when the value of *name\_or\_address* is first added to the interface or when the interface comes up. The state remains tentative until duplicate address detection (DAD) finishes, when the interface is no longer tentative. Tentative addresses are not used as source addresses.

▶ `-deprecated`

CYCLE

Specifies whether the state of the *name\_or\_address* argument is deprecated, when *address* is of type IPv6. A deprecated address is used only as a last resort. By default (the value `no`), *name\_or\_address* is not deprecated. This argument is relevant only when adding an IPv6 address.

▶ `-auto_config`

CYCLE

Specifies whether the value of the *name\_or\_address* argument is an auto-configurable IPv6 address. By default (the value `no`), *name\_or\_address* is not auto-configurable. This argument is relevant only when adding an IPv6 address.

▶ `-prefix_lifetime number`

Specifies the time, in seconds, before an IPv6 *name\_or\_address* is deprecated. The value you specify must be less than the value you specify for the `-valid_lifetime` argument. By default, this argument has no value; that is, *name\_or\_address* is never deprecated. This argument is relevant only when adding an IPv6 address.

- ▶ `-valid_lifetime number`  
 Specifies the time, in seconds, that *name\_or\_address* remains valid. By default, this argument has no value; that is, *name\_or\_address* remains valid indefinitely. This argument is relevant only when adding an IPv6 address.
  
- ▶ `-accept_router_adv value` CYCLE  
 Specifies whether the interface accepts IPv6 router advertisements. Values are *unchanged* (the default), *yes*, and *no*. By default, the interface takes on the system default for accepting IPv6 router advertisements, as the interface is added; otherwise, the interface retains the current value.
  
- ▶ `-add_default_routers value` CYCLE  
 Specifies whether detected routers are added to the default router table. Values are *unchanged* (the default), *yes*, and *no*. By default, the interface takes on the system default for adding detected routers to the default router table, as the interface is added; otherwise, the interface retains the current value.
  
- ▶ `-ip6_disabled value` CYCLE  
 Specifies whether all IPv6 network communications are disabled on the interface, while still allowing you to define addresses. Values are *unchanged* (the default), *yes*, and *no*. With the value *no*, *ifconfig* performs duplicate address detection. By default, the interface takes on the system default, as the interface is added; otherwise, the interface retains the current value.
  
- ▶ `-neighbor_detect value` CYCLE  
 Enables (the value *yes*) or disables (the value *no*) Neighbor Unreachability Detection. By default, the interface takes on the system default, as the interface is added; otherwise, the interface retains the current value (the value *unchanged*).
  
- ▶ `-prefer_source value` CYCLE  
 Specifies whether addresses on the interface are marked as a preferred source for IPv6 outgoing packets. The value *yes* marks addresses on the interface as a preferred source. By default (the value *no*), addresses on the interface are not marked as a preferred source when adding a new interface; otherwise, the interface retains the current value (the value *unchanged*).
  
- ▶ `-auto_linklocal value` CYCLE  
 Enables (the value *yes*) or disables (the value *no*) link-local addresses to be configured automatically. By default, the interface takes on the system default, as the interface is added; otherwise, the interface retains the current value (the value *unchanged*).
  
- ▶ `-euid64` CYCLE  
 Enables the command to replace the low-order 64 bits of the IP address specified by the *name\_or\_address* argument with the low-order 64 bits of the link-local address. Specify *yes* only if you also specify *yes* for the `-ip6` argument and if the *name\_or\_address* argument is an IPv6 address. By default (the value *no*),

the command does not replace the low-order 64 bits of the IP address specified by the *name\_or\_address* argument.

## Explanation

The `ifconfig` command enables you to add, delete, check, or change the status of a network interface once the STCP stack is running on the module. It also enables you to get the status of all configured STCP network interfaces and to add an additional IP address to a network interface.

You do not need to specify an interface when you issue the `ifconfig` command. If you do not specify an interface, the command lists all interfaces, as when specifying the `-all` argument. When listing interfaces, the command displays all interface names and aliases, as specified by the `-ipv4` and `-ipv6` arguments. If you specify an interface but not the `-add`, `-delete`, or `-alias` argument, the command displays information about only the specified interface.

You can specify a prefix length by including `/N` after the name in the *name\_or\_address* argument. *Prefix* refers to the combination of an IP address with a netmask (or prefix length). (Including `/N` thereby provides an alternative method of specifying a netmask.) A prefix defines the addresses within a subnet. Zero or more prefixes can be associated with an interface. You can add the first prefix with the `-add` argument. You can add subsequent prefixes with the `-add` and `-alias` arguments.

When you specify the `-add` argument and you do not specify the `-alias` and `-delete` arguments, the `-ipv4` and `-ipv6` arguments control whether IPv4, IPv6, or both are enabled on the interface.

You can configure an interface without specifying its address. You can configure the interface and then later add addresses using the `-alias` argument. You do not need to specify an address when enabling IPv6. The `ifconfig` command assigns the interface a link-local address and obtains an IP address from the router if stateless auto-configuration is enabled in the router.

When you specify the `-add` and `-alias` arguments, the value you specify for the *name\_or\_address* argument can be an IPv6 prefix, an IPv4 prefix, or an IPv4 address, as determined by the `-ipv4` and `-ipv6` arguments.

You can specify the following arguments only when you add an IPv6 address; that is, when you also specify the `-add` argument, with or without the `-alias` argument, and *name\_or\_address* is an IPv6 address:

<code>-anycast</code>	<code>-auto_configure</code>	<code>-euid64</code>
<code>-tentative</code>	<code>-prefix_lifetime</code>	
<code>-deprecated</code>	<code>-valid_lifetime</code>	



You can specify the following arguments when you add an IPv6 address (-add is yes), or when you specify an interface and the value for the -all, -add, and -delete arguments is no:

-ipv6	-default_interface
-accept_router_adv	-neighbor_detect
-add_default_routers	-prefer_source
-ip6_disabled	-auto_linklocal

The -broadcast\_type argument applies only when adding an IPv4 address (with or without -alias) and is ignored for IPv6 addresses.

You can specify the -description argument when adding an address (-add is yes), specifying an interface, and when -all, -add, and -delete are no.

Note that you can delete addresses using -delete with -alias.

Each permanent interface definition must appear in an ifconfig command line in the start\_stcp.cm command macro to ensure that each STCP logical network interface is configured at each bootload. You can add network interfaces on the fly by issuing ifconfig command lines from command level. The command uses the information you specify to notify STCP of the interface addition or deletion.

For an interface already configured for STCP, issue this command from command level to check status or change the data-transfer state of the interface to up or down. However, **do not** use this command from command level to only change the settings of the flags, IP address, or subnet mask associated with an existing (already configured) network interface. To change the setting of a flag, IP address, or subnet mask associated with an existing interface, delete the interface and add it again with the new information. If you do not need to make the change immediately, simply add the appropriate information to the ifconfig command line for the interface in the start\_stcp.cm command macro.

Before you issue the command to add (or delete) a network interface, make sure that a device entry exists for the network interface in the activated devices.table file. To add the interface, create a device entry in the devices.tin file, re-create the devices.table file and place it in the (master\_disk)>system>configuration directory, then issue the configure\_devices command to put the new devices.table file into effect.

When you delete an interface with the `-delete` argument of the `ifconfig` command, all routes with a gateway address that uses that interface are automatically deleted and the system sends a message similar to the following to the system error log:

```
14:46:23 deleting stale route 11.1.0.0/255.255.255.0
14:46:23 deleting stale route 11.2.0.0/255.255.255.0
.
.
.
```

When the *name\_or\_address* argument is an IPv4 address, specify the `-netmask` argument to inform STCP about the correct network (subnet) mask for the network interface. (For an IPv6 address, specify a network mask by including `/N` after the name in the *name\_or\_address* argument, as described earlier in this Explanation.) If you do not specify a mask, the command determines the network mask from the IP address (for example, if you issue the command with the IP address of `172.16.2.10` and do not specify a network mask, the command will use `0xffff0000` for the network mask and `172.16.255.255` for the broadcast address; it will not know that `.2` is a subnet). Therefore, if you are using subnets, you should specify the correct network mask. The `-forwb`, `-kalive`, `-mtu`, and `-broadcast_flag` arguments are the STCP flags associated with a network interface. If you do not specify these flags, the defaults are used (no forward broadcasting, keepalive, an MTU of 0, and a 1s type of broadcast).

Note that all IP addresses are equal in the kernel: you can specify a network mask and the `-broadcast_type` argument when also specifying the `-alias` argument. If you do not specify these arguments for an IPv4 address, the `ifconfig` command uses, as default values, the values from the first address currently defined on the interface.

When you configure a network/subnetwork using the `ifconfig` command, STCP automatically creates a route for the **local** network/subnetwork, based on that network address and subnet mask. These local routes are displayed by the command `route print`. If the modules are **not** on the same subnets, you must explicitly define routes using the `route` command. Note that only one route can be defined to a given destination.

The `-alias` argument allows you to perform *multihoming*, which is the ability to assign more than one IP address to the same physical interface. The interface does not need to be up in order to add or delete an alias IP address.

#### NOTE

When you specify the `-add` and `-alias` arguments together, the `ifconfig` command ignores arguments that only apply to adding an interface. Similarly, the `ifconfig` command ignores irrelevant arguments when

you specify the `-delete` and `-alias` arguments together.

## Examples

This section provides examples of the `ifconfig` command.

### Example 1: Adding an Interface

The following `ifconfig` command adds the network interface `#enet.m1.10-3` to the module.

```
ifconfig #sdlmux.10.7.2 172.16.3.15/24 -add
```

The command verifies the addition by returning the following output.

```
Adding interface %s#sdlmux.10.7.2 with IP address 172.16.3.15/24
%s#sdlmux.10.7.2: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
KEEPALIVE, MTU:1500>
ether: 0000a8 483385
172.16.3.15 netmask 0xffffffff broadcast 172.16.3.255
fe80::200:a8ff:fe48:3385%12/64 tentative
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

### Example 2: Checking Interface Status

The following examples show how to check the status of the interface by issuing the `ifconfig` command with only the device name of the interface.

The following example shows output for the device `#sdlmux.10.7.2`:

```
ifconfig #sdlmux.10.7.2

%s#sdlmux.10.7.2: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
KEEPALIVE, MTU:1500>
ether: 0000a8 483385
172.16.3.15 netmask 0xffffffff broadcast 172.16.3.255
fe80::200:a8ff:fe48:3385%12/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

The following example shows output for the device `%s1#sdlmux.11.2.1`:

```
ifconfig %s1#sdlmux.11.2.1

%s1#sdlmux.11.2.1: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
KEEPALIVE, MTU:1500>
ether: 0000a8 44bd4b
10.0.2.122 netmask 0xffffffff broadcast 10.0.2.255
fe80::200:a8ff:fe44:bd4b%6/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

You can interpret the output as follows:

- UP indicates that the network interface is administratively configured to be operational.
- BROADCAST indicates that a broadcast address is set.
- RUNNING indicates the interface's operational line state (that is, the interface is actively handling communications I/O).
- NOFORWARDBROADCAST indicates that the forward broadcast flag is disabled.
- KEEPALIVE indicates that the keepalive option is enabled.
- MTU indicates the configured MTU or packet size.

### Example 3: Checking the Status of All Configured Interfaces

The following example shows how to check the status of all interfaces configured to run STCP.

```
ifconfig

Number of interface(s) under IP: 3

%swsle#loop.m110: <UP, LOOPBACK, RUNNING, MTU:16384>
ether: 000000 000000
127.0.0.1 netmask 0xff000000
::1/128
fe80::1%1/64
nd6 options: add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.8.3: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
KEEPALIVE, MTU:9198>
ether: 0000a8 453385
10.0.244.110 netmask 0xffffffff00 broadcast 10.0.244.255
fd73:738c:286:f4::6e/64
fd73:738c:286:f4:200:a8ff:fe45:3385/64 auto_configure
fe80::200:a8ff:fe45:3385%7/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.7.2: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
KEEPALIVE, MTU:1500>
ether: 0000a8 483385
172.16.3.15 netmask 0xffffffff00 broadcast 172.16.3.255
fe80::200:a8ff:fe48:3385%12/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

### Examples 4 and 5: Changing the State of an Interface

The following example shows how to temporarily change the state of the network interface to down to temporarily disable data transfer but still keep the network interface configured for STCP.

```
ifconfig #sdlmux.10.7.2 -state down
```

The following example shows how to restore the network interface to service and resume data transfer.

```
ifconfig #sdlmux.10.7.2 -state up
```

### Example 6: Deleting an Interface

The following example shows how to delete a network interface from the list of STCP interfaces.

```
ifconfig #enet.m5.12-3 -delete
Do you really want to delete interface '#enet.m5.12-3'
: (y/n) y
Removing interface #enet.m5.12-3
```

### Example 7: Adding an Additional IP Address to an Interface

The following example shows how to add an additional IP address to an interface.

```
ifconfig #sdlmux.10.7.2 2014:1234:5678:9abc::99/64 -add -alias
```

```
Adding IP address 2014:1234:5678:9abc::99/64 to interface
%swsle#sdlmux.10.7.2
```

```
%swsle#sdlmux.10.7.2: <UP, BROADCAST, RUNNING,
NOFORWARDBROADCAST, KEEPALIVE, MTU:1500>
ether: 0000a8 483385
172.16.3.15 netmask 0xffffffff broadcast 172.16.3.255
2014:1234:5678:9abc::99/64 tentative
fe80::200:a8ff:fe48:3385%12/64
nd6 options: use_router_adv add_def_rtr neighbor_detect
auto_linklocal
```

### Example 8: Deleting an Alias IP Address

The following example shows how to delete an alias IP address.

```
ifconfig #sdlmux.10.7.1 192.168.99.9 -delete -alias
```

The command asks you to confirm the deletion:

```
Do you really want to delete IP address '192.168.99.9' (y/n) y
```

```
Deleting IP address 192.168.99.9
```

## Related Information

See “[Modifying and Executing the start\\_stcp.cm Command Macro](#)” on page 6-4, which describes the `start_stcp.cm` command macro. For more information about the network interfaces supported by STCP on ftServer systems running OpenVOS, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

## IP\_forwarding

**Privileged**

### Purpose

The `IP_forwarding` command displays or changes the current setting for the IP forwarding feature, which is relevant only if the module has multiple network interfaces configured for STCP. This command supports only the IPv4 protocol.

### Display Form

```
----- IP_forwarding -----
setting: ■
```

### Command-Line Form

```
IP_forwarding [setting]
```

### Arguments

► *setting*

**CYCLE**

The current setting for the IP forwarding feature. Specify `on` to enable IP forwarding; specify `off` to disable IP forwarding.

- When IP forwarding is enabled, the module can act as a router/gateway and forward IP datagrams to a destination on another network. If the module has multiple network interfaces connected to multiple networks or subnets, IP forwarding is enabled by default. If the module has a single network interface, you cannot enable IP forwarding. (IP forwarding has no meaning to a single network interface.)
- When IP forwarding is disabled, the module simply acts as a host and does **not** forward IP datagrams from one network to another. If you have only one network interface, IP forwarding will be disabled and the setting will remain `off` even after you issue this command to enable IP forwarding.

## Explanation

The `IP_forwarding` command enables you to display or control the IP forwarding feature for all STCP network interfaces on the module. If you issue the command without specifying a setting, it displays the current setting.

By specifying the appropriate setting, you can issue the command to manipulate the setting of the `ipForwarding` variable, which appears in the IP section of the SNMP MIB file. The module's SNMP daemon, `snmpd`, maintains the variables in its MIB file. To display the IP section of the MIB, issue the `netstat` command. The settings can be interpreted as follows:

- If IP forwarding is turned on for the module (the default for multiple network interfaces), the `ipForwarding` value in the MIB is 1.
- If IP forwarding is turned off for the module (the default for a single network interface), the `ipForwarding` value in the MIB is 2.

Note that a remote SNMP manager with the community set to the `community_private` string can also manipulate the `ipForwarding` variable on the module. Note also that the setting of this variable affects **all** STCP connections on the module; it cannot be selectively enabled or disabled on a per-network interface basis.

## Examples

The following `IP_forwarding` command disables the IP forwarding feature and configures the module (which has multiple network interfaces) as a host only.

```
IP_forwarding off
```

## Related Information

See the description of `snmpd` in [Chapter 8](#) for more information about `snmpd` and the SNMP variables.



## ip\_pair\_admin

### Purpose

The `ip_pair_admin` command enables you to control path MTU discovery on network paths between two IP addresses. The pair of IP addresses consists of a local IP address on the current module and the IP address of the remote host.

Any user can issue this command to display pairs of IP addresses; however, you must be a privileged user to make changes in path MTU discovery.

### Display Form

```
----- ip_pair_admin -----
action:          list
remote_address:
local_address:
-mtu:
-network:
-numeric:        no
-interval:
-wide:           no
-ipv4:           yes
-ipv6:           yes
-module:
```

### Command-Line Form

```
ip_pair_admin [action]
               [remote_address]
               [local_address]
               [-mtu number]
               [-network string]
               [-numeric]
               [-interval number]
               [-wide]
               [-no_ipv4]
               [-no_ipv6]
               [-module module_name]
```

## Arguments

► *action*

CYCLE

**Required**

Specifies the action that you want to perform on the pair of IP addresses. Though any user can issue the `ip_pair_admin` command with the `list` action, you must be privileged to specify actions that make changes.

Specify one of the following actions:

- `list`—Displays pairs of IP addresses between which the module (specified by the `-module` argument) has an active network path. Any user can issue the `ip_pair_admin` command with the `list` action.
- `add`—Adds a pair of IP addresses to an internal cache of IP addresses.  
With `add`, you can use the `-mtu` argument to specify an MTU for the network path between the added pair of IP addresses.
- `set_mtu`—Sets the MTU for the network path between *remote\_address* and *local\_address* that already exists in the database.
- `reset_mtu`—Resets the MTU for the network path between *remote\_address* and *local\_address*. This action is identical to specifying the `-mtu 0` argument.

► *remote\_address*

Specifies the IP address of the remote host. If you specify *remote\_address* and `list` for the *action* argument, note the following information:

- You can use CIDR notation by specifying `/N` after the name, where *N* is referred to as the prefix length and indicates the number of high-order 1 bits in the netmask.
- The command output displays only pairs of IP addresses whose remote IP address matches *remote\_address*.

► *local\_address*

Specifies the IP address of an SDLMUX-group interface on the module specified by the `-module` argument. If you specify *local\_address* and `list` for the *action* argument, note the following information:

- You can use CIDR notation by specifying `/N` after the name, where *N* is referred to as the prefix length and indicates the number of high-order 1 bits in the netmask.
- The command output displays only pairs of IP addresses whose local IP address matches *local\_address*.

- ▶ `-mtu number`  
Specifies an unsigned integer limit for the MTU used when sending packets from *remote\_address*. It is applied when you also specify `add` or `set_mtu` for the *action* argument. Specifying the value 0 for *number* is identical to specifying `-reset_mtu` for the *action* argument. See the [Explanation](#) for more information on specifying an MTU value.
- ▶ `-network string`  
Specifies an IP address and option mask that determines the IP pairs displayed in the command output. The value functions as a filter that limits output when you also specify `list` for the *action* argument. The `ip_pair_admin` command displays all IP pairs that have a local or remote address matching *string*. Specify the *string* value in CIDR notation.
- ▶ `-numeric` CYCLE  
Disables the command from converting IP addresses to domain names, so that the output displays only IP addresses. By default (the value `no`), the output includes domain names converted from IP addresses.
- ▶ `-interval number`  
Specifies the amount of time, in seconds, between updates of the displayed data. If you specify this argument, the updates continue until you type a break sequence to break out of the command. For information about keys you can press to alter the display, see [Table 9-5](#) in the `netstat` command description.
- ▶ `-wide` CYCLE  
Enables the command output to include additional columns displaying statistics. By default (the value `no`), the command includes a minimal number of columns displaying statistics.
- ▶ `-no_ipv4` CYCLE  
Disables the command's search for IPv4 addresses as well as the display of information about IPv4 address pairs. By default (the value `yes`), the command searches for IPv4 addresses and displays information about IPv4 address pairs.
- ▶ `-no_ipv6` CYCLE  
Disables the command's search for IPv6 addresses as well as the display of information about IPv6 address pairs. By default (the value `yes`), the command searches for IPv6 addresses and displays information about IPv6 address pairs.
- ▶ `-module module_name`  
Specifies the module on which to list or modify IP pairs.

## Explanation

The `ip_pair_admin` command enables you to control path MTU discovery over one or more network paths between an SDLMUX-group interface (specifically, the IP

address for the interface) on the module specified by the `-module` argument and the IP address of a remote host.

The path MTU discovery algorithm probes remote networks to determine if communication can occur with an MTU value that is larger than the default value. The IP layer includes the state for path MTU discovery. The IP layer also includes a cache of IP address pairs (local and remote) that are actively communicating. This cache enables STCP to assign the most unique identifiers to IP packets, in order to ensure reliable fragmentation, because communication occurs at widely varying network speeds on different subnets. This cache is also used to maintain the state for path MTU discovery.

Setting the MTU (that is, specifying a value for the `-mtu` argument) disables STCP from probing to determine if it can use a larger MTU. Setting the MTU also locks in memory the cached pair of IP addresses. A cached pair of IP addresses is typically freed when protocols finish using the path. Specifying the value 0 for `-mtu` re-enables probing and allows that cached pair of IP addresses to be removed from the cache when all the uses go away.

The default value for the `-mtu` argument is zero (no limit). The actual maximum MTU used for sending packets is the minimum value of the following:

- The maximum MTU supported by the hardware. This value is fixed by the hardware.
- The MTU specified by the `ifconfig` command (the default is 1500)
- The MTU limit for the route (if it exists)
- The MTU limit on the IP pair (if it exists). This value can be set and/or reduced by path MTU discovery.

To control the IP pairs displayed in command output, you can specify values for the following arguments:

- `remote_address`
- `local_address`
- `-network`
- `-no_ipv4`
- `-no_ipv6`

The following arguments enable you to control the format of the command output:

- `-numeric`
- `-interval` (which also creates an interactive display)
- `-wide`

## **Related Information**

See descriptions of the [ping](#) and [route](#) commands.

## **ip6addrctl**

The `ip6addrctl` command configures the IPv4 and IPv6 address selection policy.

The `ip6addrctl` command controls which IPv6 addresses are preferred as source addresses when an application does not specify a source address. The command also determines the sort ordering for addresses returned by the `getaddrinfo` function; that is, the command determines which types of addresses are preferred when the DNS returns multiple addresses for a domain name. Use of the command would typically be necessary when the DNS returns both IPv4 and IPv6 addresses.

For more information on the `ip6addrctl` command, see online BSD documentation (for example, <http://www.freebsd.org>).

For information on POSIX.1 functions available for OpenVOS, including `getaddrinfo`, see the *OpenVOS POSIX.1 Reference Guide* (R502).

## netstat

### Purpose

The `netstat` command enables you to display information about the status of a network interface, STCP connections, STCP protocols, and STCP routes. If you specify the `netstat` command without any arguments, it displays active TCP connections as well as bound UDP sockets.

To save information displayed by the `netstat` command, use the `start_logging` and `stop_logging` commands. For detailed descriptions of these commands, see the *OpenVOS Commands Reference Manual (R098)*.

### Display Form

```
----- netstat -----
-interface:          █
-protocol:
-network:
-PCB_addr:           no
-all_sockets:        no
-listeners:          no
-datagram:           no
-routing:            no
-statistics:         no
-numeric:            no
-interval:           0
-multicast:          no
-64:                 yes
-display_zeros:      no
-columnar:           yes
-show_all_digits:    no
-shorten_path_names: yes
-wrap:               yes
-batch_line_length:  80
```

## Command-Line Form

```
netstat [-interface string]  
        [-protocol string]  
        [-network string]  
        [-PCB_addr]  
        [-all_sockets]  
        [-listeners]  
        [-datagram]  
        [-routing]  
        [-statistics]  
        [-numeric]  
        [-interval number]  
        [-multicast]  
        [-no_64]  
        [-display_zeros]  
        [-no_columnar]  
        [-show_all_digits]  
        [-no_shorten_path_names]  
        [-no_wrap]  
        [-batch_line_length number]
```

## Arguments

► **-interface *string***

Displays detailed information about the specified network interface that is configured to run STCP. The output associated with this argument depends on the type of network interface. To specify a network interface, type the number sign (#), followed by the OpenVOS device name of a network interface that has been configured with the command `ifconfig -add` or is the active or standby member of an SDLMUX pair that has been configured with the command `ifconfig -add`. If you also specify the `-64` argument, you can specify an asterisk (\*) for *string*.

The command displays different information, depending on the type of device name you specify for *string*, as follows:

- If you specify the name of an un-partnered interface, the command displays detailed information about the device.
- If you specify the name of an SDLMUX device (that is, an SDLMUX-group interface), the command displays information about both the active and standby devices.



- If you specify the name of either an active or standby device that is part of an SDLMUX pair, the command displays detailed information about the specified device.

► `-protocol string`

CYCLE

For *string*, specify one of the following values:

<code>all</code>	<code>igmp</code>
<code>af_local</code>	<code>ip</code>
<code>af_unix</code>	<code>raw</code>
<code>icmp</code>	<code>tcp</code> (or for more information, <code>tcp/ext</code> )
<code>if</code> (or its equivalent, <code>if+arp</code> )	<code>udp</code>

When you specify `tcp`, `udp`, or `af_unix` for `-protocol` without specifying the `-statistics` argument, `netstat` displays a list of the connections associated with the specified protocol. When you specify `all` for this argument and the `-statistics` argument, the command displays a list of the statistics associated with the specified protocols.

If the `netstat` command does not recognize the protocol you specify, the command generates an error message. By default, the command does not display a list of connections associated with a protocol or a list of protocol statistics.

► `-network string`

Displays only information that is relevant to the network specified by *string* (for example, 172.16.2.0). Note that *string* must be a network address (IPv4 or IPv6). Use this argument when you want to request information associated with only one network/subnet. For example, specify this argument with the `-all_sockets` argument to verify all socket connections associated with a specified network. By default, the `netstat` command displays information for all networks.

► `-PCB_addr`

CYCLE

Displays an additional column, `PCB`, to identify the address of the Protocol Control Block (PCB). By default, the PCB information is not displayed. This PCB address is used for debugging purposes only.

► `-all_sockets`

CYCLE

Displays additional information about the connection states for the STCP servers. If a server is just listening and has no active connection yet, an asterisk (\*) appears in the output. By default, the `netstat` command does not display the connection states of the servers.

▶ `-listeners`

CYCLE

Displays information about listening sockets (TCP and AF UNIX). Listening sockets are sockets waiting for an incoming connection. (In contrast, the `-all_sockets` argument displays information about listening sockets and connected sockets.) By default, the `netstat` command does not display information about listeners.

▶ `-datagram`

CYCLE

Displays information about only SOCK\_DGRAM sockets (IP and AF UNIX). By default, the command displays information about all relevant sockets.

▶ `-routing`

CYCLE

Displays the routing table. The routing table is empty until routes are established, so if you issue the command and the table is empty, the command will reflect that. If you specify both this argument and the `-statistics` argument, the command displays routing statistics instead of the routing table. The routing information is always displayed in numeric form. By default, the `netstat` command does not display routing information.

▶ `-statistics`

CYCLE

When issued without additional arguments, displays statistics for all of the protocols. Specify this argument with the `-protocol` argument to display protocol statistics for a particular protocol. Specify this argument with the `-routing` argument, to display routing statistics instead of protocol statistics. By default, the `netstat` command does not display statistics.

▶ `-numeric`

CYCLE

Displays host and network addresses in numeric (standard dot-notation) form. When displaying information about connections, the `netstat` command tries to resolve an address and display its corresponding symbolic name, unless you explicitly specify the `-numeric` argument. Routing table information is always displayed in numeric form.

▶ `-interval number`

Specifies the amount of time, in seconds, between updates of the displayed data. Specify this argument when you are requesting statistics, connection information, interface information, protocol information, or routing information. If you specify this argument, the updates continue until you type a break sequence to break out of the command. Specify the `-numeric` argument with `-interval` to conserve CPU-processing time.

For information about specifying the `-interval` argument with the `-64` argument, see the description of the `-64` argument.

▶ `-multicast`

CYCLE

Displays group membership information (that is, information about which IP multicast addresses are registered to a specific interface). If no IP multicast addresses are registered to a specific interface, no IP multicast addresses will

appear in the output. By default, the `netstat` command does not display group membership information.

► `-no_64`

CYCLE

Suppresses an implementation of `netstat` that uses the 64-bit counter API of the kernel. The default value `yes` invokes an implementation of `netstat` that uses the 64-bit counter API, which affects the following:

- The display is more readable. For example, all data for one protocol appears in one section under a descriptive heading, rather than in single lines. In addition to providing more information, this display allows more horizontal space for columns and adjusts column size, based on display width.
- The value of the `-suppress_zeros` argument is ignored; instead, use the `-display_zeros` argument.
- When you specify the value `tcp/ext` for the `-protocol` argument as well as the `-64` argument, the display includes extended statistics with the `stcp` statistics. The extended statistics include 22 reset meters as well as the `STCP` parameters and meters that appear when you issue the `list_stcp_params` and `stcp_meters` requests of `analyze_system`.
- When you specify the `-64` and `-network string` arguments, the value you specify for `string` can be one of the following:
  - CIDR notation to specify an IPv4 or IPv6 network
  - A vertical bar (|) followed by a netmask

The value applies to all tables displayed that contain an IP address.

- When you specify the `-64` argument with the `-interval` argument, you can alter the display by pressing one or more keys. The `-interval` argument specifies the number of seconds between updates of the displayed data; in addition, the display appears in a screen mode that enables you to scroll within the display, even while it is being refreshed at the specified interval. Updates continue until you break out of the command or exit by typing the letter `q` (or one of its equivalents described in [Table 9-5](#)). If you are using SSH, this mode tracks the display being resized.

You can alter the display by pressing one of the keys listed in [Table 9-5](#).

**Table 9-5. Keys That Change the `-interval` Display** (Page 1 of 2)

Key(s)	Change
0	Pressing the digit zero causes all numbers to display with three significant digits and scale factors (for example, 1.99K). This display is the default.

**Table 9-5. Keys That Change the -interval Display** (Page 2 of 2)

Key(s)	Change
9	Pressing the digit nine causes all numbers to display with all significant digits (for example, 1990).
a A	Pressing the letter a (or A) toggles on or off the display of numbers aligned by their decimal points (in three-digit mode).
p P	Pressing the letter p (P or any keystroke sequence of your terminal that scrolls the display up) moves the cursor up.
b B	Pressing the letter b (B or any keystroke sequence of your terminal that displays the previous screen) displays the previous screen of information.
c C	Pressing the letter c (or C) toggles on or off the incremental display of all counters.
n N	Pressing the letter n (N or any keystroke sequence of your terminal that displays the next screen) moves the cursor down.
f F	Pressing the letter f (F, the space bar, or any keystroke sequence of your terminal that displays the next screen) displays the next screen of information.
g G	Pressing the letter g (or G) toggles on or off the incremental display of all gauges.
q Q	Pressing the letter q (Q or any keystroke sequence of your terminal that cancels, quits, or exits the form) exits the program.
s S	Pressing the letter s (or S) toggles on or off the inverse lookup for IP addresses. This display is an interactive version of the display provided by the <code>-numeric</code> argument.
z Z	Pressing the letter z (or Z) toggles on or off the display of counters whose value is zero. This display is an interactive version of the display provided by the <code>-display_zeros</code> argument.

- ▶ `-display_zeros` CYCLE  
Displays statistics of the `-64` argument whose values are zero. If you specify the `-display_zeros` argument, you must also specify the `-64` argument. By default, the command `netstat -64` does not display statistics whose values are zero.
- ▶ `-no_columnar` CYCLE  
Displays statistics in lines rather than in columns. By default (the value `yes`), `netstat` displays most tabular output in columns.
- ▶ `-show_all_digits` CYCLE  
Displays all significant digits when specified with the `-64` argument instead of the default display of three significant digits and scale factors. This display is identical

to the display that appears after issuing `netstat -interval` and pressing 9 (see [Table 9-5](#)). By default, the command `netstat -64` displays three significant digits and scale factors.

- ▶ `-no_shorten_path_names` CYCLE  
Displays full path names. By default (the value `yes`), the command displays shortened path names by, for example, removing the current directory.
- ▶ `-no_wrap` CYCLE  
Formats non-interactive output so as to not truncate fields. By default (the value `yes`), the command generates lines that wrap. Non-interactive output occurs when the `-interval` argument is not used and there is no login terminal.
- ▶ `-batch_line_length number`  
Specifies the line length when running in batch mode and `-no_wrap` is set to `yes`.

## Explanation

The `netstat` command enables you to display information about the following:

- connections and connection states
- a network interface configured for STCP
- statistics for protocols such as ARP, IP, TCP, UDP, ICMP, IGMP, and AF UNIX
- the routing table and routing statistics

You should issue `netstat` to monitor your configuration periodically. You should record statistics once per day or even once per hour, if possible. Look for values that are typically zero and that are now nonzero.

In general, the `netstat` command enables you to display pertinent STCP information in one of several formats. The formats can be summarized as follows:

- Format 1 displays connection information, either a list of the active user connections or the connection status associated with all ports, including those used by listening servers if you specify the `-all_sockets` argument. Processes that have sockets that are listening will only be displayed in this manner. Omitting the connection status of all listening servers from the display makes the display more concise. Use the `-protocol` argument to list the connections associated with TCP, UDP, or AF UNIX. You can also specify the arguments `-PCB_addr`, `-listeners`, and `-datagram` with this format. Specify the `-network` argument if you have multiple network/subnet connections and want to display only the connection information that applies to one of the networks/subnets.
- Format 2 displays network interface information, for a network interface specified with the `-interface` argument. When displaying detailed information regarding network interfaces, the `netstat` command uses a format that applies to the

specific type of network interface (for example, Ethernet). If you are using partnered network interfaces, note that the device name you specify (the one you specified when configuring the representative logical interface) refers to both partners at the STCP level. The command displays the MAC statistics associated with both the active and standby devices.

- Format 3 displays protocol statistics, for one or all of the STCP protocols:

<code>af_local</code>	<code>ip</code>
<code>af_unix</code>	<code>raw</code>
<code>icmp</code>	<code>tcp (or tcp/ext)</code>
<code>if (or its equivalent, if+arp)</code>	<code>udp</code>
<code>igmp</code>	

You use the `-statistics` argument alone (or the `-protocol all` argument with the `-statistics` argument) to display the statistics for all protocols. (The `-protocol` argument on its own does not display the statistics; it displays connection information.) The `-statistics` argument displays statistics for each protocol and errors detected in the protocol software. These statistics are recorded from the time the STCP software is loaded. Use the `-statistics` argument to display all statistics. When issued without the `-64` argument, the command displays all fields (including those that have a value of 0) in the same order each time you issue the command. When issued with the `-64` argument, you must also use the `-display_zeros` argument to display fields that have a value of 0.

- Format 4 displays routing information. A routing table is a database of routes. You use the `-routing` argument to display the routing table. Each route entry identifies a destination, router/gateway, and subnet mask. Optionally, a route can also specify information such as whether an ICMP redirect request redirected transmission to another router/gateway and how long until the redirect route expires. Use the `-routing` argument and the `-statistics` argument to display routing statistics.
- Format 5 displays group membership information. You use the `-multicast` argument to display group membership information.

When you specify the `-64` and `-network` arguments, `netstat` displays routing information as well as group membership information; specifically, the command output includes information from any table that contains one or more IP addresses.

The following sections provide additional information about the `netstat` command output:

- [Information in netstat Command Output](#)
- [Sample Output of the netstat Command](#)

## Information in netstat Command Output

The following tables provide information about fields, statistics, variables, and messages that can appear in output of the `netstat` command.

- [Table 9-8](#) describes fields that identify connections.
- [Table 9-9](#) describes TCP protocol states.
- [Table 9-10](#) lists fields displayed for MAC statistics.
- [Table 9-11](#) lists IF statistics.
- [Table 9-12](#) lists ICMP statistics with overall IPv4 and IPv6 counts.
- [Table 9-13](#) lists examples of ICMP message types.
- [Table 9-14](#) lists TCP statistics displayed with `tcp/ext`.
- [Table 9-15](#) lists UDP statistics.
- [Table 9-16](#) lists AF UNIX statistics.
- [Table 9-17](#) lists problems indicated by fields in `netstat` statistics.

Output of the `netstat` command can include fields, statistics, variables, or messages not included in the tables listed above. For such information, search RFCs listed in [Table 9-6](#) and MIB files listed in [Table 9-7](#). You can download RFCs from <http://www.rfc-editor.org/>.

**Table 9-6. RFCs with netstat Command Output Information**

RFC	Common Name
RFC-1213	Legacy network MIBs
RFC-2863	IF MIB
RFC-3635	Etherlike MIB
RFC-4022	TCP MIB
RFC-4113	UDP MIB
RFC-4292	IP Forward MIB
RFC-4293	IP MIB
RFC-4836	Medium Attachment Unit (MAU) MIB
RFC-5519	Multicast Group Membership Discover (MGMD) MIB

[Table 9-7](#) lists OpenVOS management information base (MIB) files, which are machine-readable database files that reside in the `(master_disk)>system>vos_mibs` directory. MIB files, though often associated

with SNMP, can be used within other network management models. OpenVOS uses its MIB files to describe non-standard variables in STCP. To view the MIB files, use a MIB-parsing utility or a text editor (for example, Emacs). Use the `access_info_monitor` command to display actual values from the kernel's MIB database.

**Table 9-7. OpenVOS MIB Files in (master\_disk)>system>vos\_mibs**

OpenVOS MIB Files
VOSAFLOCAL-MIB.mib
VOEXTEN-MIB.mib
VOSNETWORK-MIB.mib

Table 9-8 describes fields that appear in the output of the `netstat` command to identify connections.

**Table 9-8. Fields in netstat That Identify Connections (Page 1 of 2)**

Field	Description
Proto	The name of the protocol used by the connection.
Recv	The total amount of data, in bytes, received on the connection.
Recv Q	The amount of data, in bytes, received but not passed to the user because of flow control.
Sent	The total amount of data, in bytes, sent on the connection.
Send Q	The amount of data, in bytes, waiting to be transmitted. For TCP, this includes data sent but not yet acknowledged.
Local Address	<p>The IP address of the local host, and the port number the connection is using. The host and port number are separated by a colon (:). If the local host has an entry in the <code>hosts</code> database file or the DNS, its corresponding host name is displayed instead of its IP address. Note that when using a DNS server which is currently not working, the command may take between 30 seconds and several minutes to display each line while it waits for the DNS query to time out. If the IP address is not bound (set to 0.0.0.0), the IP address is shown as an asterisk (*). To bypass DNS name resolution, use the <code>-numeric</code> argument.</p> <p>If the port has a service name defined for it in the <code>services</code> database file, the service name is displayed instead of the port number. If the port is not yet established, the port number is shown as an asterisk (*).</p>



**Table 9-8. Fields in netstat That Identify Connections** (Page 2 of 2)

Field	Description
Foreign Address	<p>The IP address and port number of the remote host to which the socket is connected. Note that when using a DNS server which is currently not working, the command may take between 30 seconds and several minutes to display each line while it waits for the DNS query to time out. If the IP address is not bound (set to 0.0.0.0), the IP address is shown as an asterisk (*). To bypass DNS name resolution, use the <code>-numeric</code> argument.</p> <p>If the port has a service name defined for it in the <code>services</code> database file, the service name is displayed instead of the port number. If the port is not yet established, the port number is shown as an asterisk (*).</p>
State	The TCP connection state. (See <a href="#">Table 9-9</a> for a list of the connection states.)

As defined by RFC 793, the TCP protocol of a TCP/IP connection is in one state at any given time. The state indicates current activity between the client and the server. Subsequent activity causes a transition to another state.

[Table 9-9](#) describes the TCP protocol states. See RFC 793 to examine the TCP state diagram.

**Table 9-9. TCP Protocol States** (Page 1 of 2)

Protocol State	Description
closeWait	Side B <sup>†</sup> enters this state when it receives a shutdown message from side A. Side B begins closing its side of the connection and sends side A an acknowledgment reply.
closed	The socket is not in use; the client and the server are inactive.
closing	Side A <sup>†</sup> enters this state when it sends a shutdown message to side B and receives a shutdown message from side B (side A and side B begin shutdown at approximately the same time). In other words, side A sends a shutdown message to side B and begins closing its side of the connection; side B sends a shutdown message to side A and begins closing its side of the connection before it receives the shutdown message from side A.
established	The client enters this state when it receives the server's acknowledgment reply and sends the server an acknowledgment reply. The server enters this state when it receives the client's acknowledgment reply and data transfer proceeds.
finWait1	Side A <sup>†</sup> enters this state when it sends a shutdown message to side B and begins closing its side of the connection.

**Table 9-9. TCP Protocol States** (Page 2 of 2)

Protocol State	Description
<code>finWait2</code>	Side A <sup>†</sup> enters this state when it receives an acknowledgment reply from side B for its shutdown message. Side A waits for a message indicating that side B has finished closing its side of the connection.
<code>lastAck</code>	Side B <sup>†</sup> enters this state after it has finished closing its side of the connection in response to a shutdown message from side A. Side B has sent side A its shutdown message and is now waiting for side A's final acknowledgment reply.
<code>listen</code>	The socket is open and the server is listening for incoming connection messages.
<code>synReceived</code>	The server enters this state after it receives the client's first message and sends the client an acknowledgment reply. The server waits to receive the client's acknowledgment of that message.
<code>synSent</code>	The client enters this state when it sends the server the first message to establish a connection and is waiting for the server's acknowledgment of that message.
<code>timeWait</code>	Side A <sup>†</sup> enters this state when it receives the final acknowledgment reply from side B. Side A finishes closing its side of the connection and sends a final acknowledgment reply to side B.

† Side A sends the first shutdown message; side B receives the message. (States during shutdown are independent of both the client and server; either the client or the server can begin the shutdown.)

[Table 9-10](#) describes fields that appear in the `MAC Statistics` section of the Ethernet output.

**Table 9-10. Fields in the MAC Statistics Section of Ethernet Output** (Page 1 of 2)

Field	Description
<code>Received frames</code>	The number of frames received that were addressed to the interface.
<code>Received multicast and broadcast frames</code>	The number of multicast and broadcast frames received.
<code>Received octets</code>	The number of octets (8-bit bytes) received, including framing octets.
<code>Transmitted frames</code>	The number of frames transmitted.
<code>Transmitted octets</code>	The number of octets transmitted, including framing octets.
<code>LAN chipset re-initialized</code>	The number of times the chip was reinitialized.

**Table 9-10. Fields in the MAC Statistics Section of Ethernet Output** (Page 2 of 2)

Field	Description
SQE error	The number of SQE test failures.
Transmit ring full	The number of times that the module sent a frame down for transmission and the card had no buffer space to hold the data.
Transmit frame discarded, late collision	The number of late collisions.
Transmit frame was deferred	The number of times a frame transmission was deferred. A deferral occurs when the card tests the carrier prior to transmission and finds that the network is already in use by another host.
Transmit frame after a single retry	The number of times that a frame transmission was tried once prior to a successful transmission. The first time, a collision occurred, and the second time, the transmission completed without problems.
Transmit frame after multiple retry	The number of times more than one retry was required but fewer than the maximum 16 retries.
Transmit frame discarded, excessive retry	The number of times a frame could not be successfully transmitted after 16 retries.
Receive frame discarded, improper framing	The number of times that a frame arrived that did not contain an integer multiple of eight bits.
Receive frame discarded, lack of buffers	The number of times that data arrived from the network but the card did not have any buffers to send it up the protocol stack.
Receive frame discarded, an overflow	See the description of the field "Receive frame discarded, lack of buffers."
Received frame discarded, congestion	See the description of the field "Receive frame discarded, lack of buffers."
Receive frame discarded, bad CRC	The number of times that a frame arrived with proper framing but a bad CRC.
Received frame discarded, bad address	The number of times that the card received a frame for an address that is not configured.

You can issue the `netstat` command with the `-statistics`, `-protocol`, and `-interval` arguments to display protocol statistics. You should obtain a baseline

reading of protocol statistics for comparison and reference. The following tables list and describe fields that report protocol statistics and related information:

[Table 9-11](#) lists interface (IF) statistics.

[Table 9-12](#) lists ICMP statistics with overall IPv4 and IPv6 counts.

[Table 9-13](#) lists examples of ICMP message types.

[Table 9-14](#) lists TCP statistics displayed with tcp/ext.

[Table 9-15](#) lists UDP statistics.

[Table 9-16](#) lists AF UNIX statistics.

[Table 9-17](#) lists problems indicated by fields in netstat statistics.

For descriptions of other statistics fields that appear in `netstat` output for protocols, see MIB files in the `(master_disk_>system>vos_mibs` directory.

[Table 9-11](#) lists and describes the fields that report interface (IF) statistics.

**Table 9-11. Fields in netstat That Report IF Statistics** *(Page 1 of 2)*

Field	Description
ifName	The textual name of the interface.
ifDescr	A textual string containing information about the interface.
ifPhysAddress	The interface's address at its protocol sub-layer.
ifHighSpeed	An estimate of the interface's current bandwidth in units of 1,000,000 bits per second. If this object reports a value of <i>n</i> , the speed of the interface is in the range of <i>n</i> -500,000 to <i>n</i> +499,999. The scale factor is applied when the value is displayed.
dot3VosFifoOverruns	The number hardware FIFO overruns. This field is an indicator of the hardware (PCI bus) performance.
dot3VosInputOverruns	The number of memory buffer overruns. This field is an indicator of the software performance.
dot3VosInputAverageQueueDepth	The average number of buffers queued in an input ring.
dot3VosInputMaxQueueDepth	The maximum number of buffers that have been queued in an input ring before being handled by an interrupt.
dot3VosOutputOverruns	The number of times output flowed off because an output ring was full.
dot3VosOutputAverageQueueDepth	The average number of buffers queued in an output ring.

**Table 9-11. Fields in netstat That Report IF Statistics** (Page 2 of 2)

Field	Description
dot3VosOutputMaxQueueDepth	The maximum number of buffers that have been queued in an output ring before being transmitted and subsequently freed by an interrupt.

Table 9-12 and Table 9-13 list ICMP fields and examples of messages that appear in output of the `netstat` command issued with the `-64` argument (you can use the `-display_zeros` argument to display all fields and message types):

- Table 9-12 lists and describes the fields that report ICMP statistics with overall counts for both IPv4 and IPv6 communications.
- Table 9-13 lists examples of the exact ICMP message types that appear in output of the `netstat` command with incoming and outgoing packet statistics.

For information on ICMP statistics and messages, see these web sites:

- <http://www.networksorcery.com/enp/protocol/icmp.htm>
- <http://www.networksorcery.com/enp/protocol/icmpv6.htm>

**Table 9-12. Fields in netstat That Report ICMP Statistics with Overall IPv4 and IPv6 Counts**

Field	Description
icmpStatsInMsgs	The number of ICMP messages that the entity received. Note that this counter includes all the messages counted by <code>icmpStatsInErrors</code> .
icmpStatsInErrors	The number of ICMP messages that the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.).
icmpStatsOutMsgs	The number of ICMP messages that the entity attempted to send. Note that this counter includes all the messages counted by <code>icmpStatsOutErrors</code> .
icmpStatsOutErrors	The number of ICMP messages that this entity did not send due to problems discovered within ICMP, such as a lack of buffers. This value should not include errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations, there may be no types of error that contribute to this counter's value.

**Table 9-13. Examples of ICMP Message Types Displayed with Incoming and Outgoing Packet Statistics**

ICMP Message Type Examples
v4Echos
v4EchoReplies
v4DestinationUnreachs
v4Redirects
v6Echos
v6EchoReplys
v6DestinationUnreaches
v6PacketTooBigs
v6NdRedirects

Table 9-14 lists and describes some of the variables that the netstat command displays when issued with the value tcp/ext for the -protocol argument. In the table, please note the following:

- For variables described in RFC-4022, the high capacity (HC) version of the standard is used when available, but the netstat output does not include HC in the name to reduce screen clutter. HC indicates a 64-bit counter in the kernel. For non-HC variables, netstat returns the low order 32 bits. See Table 1-1 for information on how to view an RFC.
- In the table, the description of some variables list values of stcp\_meters, list\_stcp\_params, or set\_stcp\_param requests of the analyze\_system subsystem. For example, the tcpVosDataSegmentsReceived variable lists “segments received in stcp\_meters,” which refers to the segments received variable listed by the stcp\_meters request of analyze\_system. For information on the analyze\_system command and its requests, see the *OpenVOS System Analysis Manual* (R073). Note, however, that the netstat command issued with the value tcp/ext for the -protocol argument provides a more complete display of STCP statistics and the access\_info\_monitor command provides more complete access to variables affecting STCP performance than any analyze\_system request.
- The list of variables in Table 9-14 is incomplete. For information about variables not listed in the table, see the RFCs listed in Table 9-6 and the OpenVOS MIB files listed in Table 9-7.

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 1 of 16)**

Variable	Description or Description Location
tcpRtoAlgorithm	RFC-4022
tcpRtoMin	
tcpRtoMax	
tcpMaxConn	
tcpVosMaxTCBs	The maximum number of TCP transmission control blocks (TCBs). The default value is 70000.
tcpVosFirstNonPrivilegedPort	The number of the first non-privileged port.
tcpVosFirstDynamicPort	The number of the first dynamic port.
tcpVosFirstSharedDynamicPort	The number of the first shared dynamic port.
tcpActiveOpens	RFC-4022
tcpPassiveOpens	
tcpAttemptFails	
tcpEstabResets	
tcpCurrEstab	
tcpInSegs	Counts all segments received, including ACKs and bad segments. See also RFC-4022.
tcpOutSegs	Counts all segments sent, except retransmissions. See also RFC-4022.
tcpRetransSegs	RFC-4022
tcpInErrs	
tcpOutRsts	
tcpVosDataSegmentsReceived	The total number of segments received. Count includes only segments with data. This variable is also <code>segments received</code> in <code>stcp_meters</code> .
tcpVosBytesReceived	The total number of bytes received. This variable is also <code>total bytes</code> in <code>stcp_meters</code> .
tcpVosSegmentsSent	The total number of segments sent. Count includes all data segments sent, including retransmissions. This variable is also <code>segments sent</code> in <code>stcp_meters</code> .

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 2 of 16)**

Variable	Description or Description Location
tcpVosTotalBytesSent	The total number of bytes sent. This variable is also total bytes in stcp_meters.
tcpVosRetransmissions	The total number of segments retransmitted when an ACK is not received in a timely fashion. This variable is also retransmissions in stcp_meters.
tcpVosDuplicateAcksReceived	The number of ACKs received without data and without increasing window size. This data indicates packet loss. This variable is also duplicate acks received in stcp_meters.
tcpVosSegmentsOutOfOrder	Number of out-of-order segments received that are inside the receive window but that do not have the expected sequence number. These segments are buffered. This variable is also slow path in stcp_meters.
tcpVosHeaderTruncated	The number of packets received that are too short.
tcpVosChkSumErrors	The number of packets received with checksum errors.
tcpVosSocketsAllocated	The number of TCB structures allocated.
tcpVosSynsDiscarded	The number of SYNs discarded for any reason.
tcpVosMaxSegmentLifetime	<p>The amount of time that a socket spends in the TIME_WAIT state. Typically, you can adjust this time from 60 seconds (the default) to a time in the range of 10 to 600 seconds. The TIME_WAIT state is entered only by the socket that closed the connection and only after receiving indication that the remote end of the connection closed its socket. The TIME_WAIT state ensures that any late-arriving packets for the connection are not confused with packets arriving for a new connection. Typically, you do not need to reduce this timer. Applications sometimes receive the error "Address in Use" when they terminate and start up again before the maximum segment lifetime (MSL) timer has expired. To prevent this error, set the SO_REUSEADDR socket option on the socket rather than reducing the MSL timer.</p> <p>This variable is also tcp_msl in list_stcp_params and set_stcp_param.</p>



**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 3 of 16)**

Variable	Description or Description Location
tcpVosLogLandPackets	<p>A switch that allows STCP to log to the system error log the number of the following types of problems:</p> <ul style="list-style-type: none"> <li>– LAND packets discarded</li> <li>– connect requests discarded during shutdown of the STCP stack</li> <li>– connect requests discarded when the value of the <code>-detect_syn_attack</code> parameter is on</li> </ul> <p>By default, this switch is on. To ensure the logging of land denial-of-service (DOS) attacks, do not set this switch set to off. This variable is also <code>log_syn_attacks</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosFinwait2Timeout	<p>The time, in seconds, to be specified for the TCP state <code>FIN_WAIT_2</code>. When a connection is closed, the closing socket sends notification to the remote socket, and after TCP acknowledges the notification, the socket is placed in the <code>FIN_WAIT_2</code> state. It remains in that state until it receives notification from the remote socket that it has closed. The socket remains in the <code>FIN_WAIT_2</code> state forever, if the remote socket does not notify it. This situation happens when the remote application hangs or is no longer processing the remote socket. Setting this parameter allows you to clean up the sockets, but technically violates the TCP protocol standard. This variable is also <code>finwait2</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosKeepaliveKeepaliveTimeout	<p>A keepalive timer that tests a connection to confirm whether the remote end is alive or has failed, and that cleans up the socket if the remote end has failed. A keepalive probe is set only if the socket has <code>SO_KEEPAIVE</code> set and if the interface used to send the probe has <code>kalive</code> set to yes. The keepalive timer is really a <i>garbage-collection</i> function used to clean up connections that have failed and have not been cleaned up in some other way. The keepalive timer is not an alternative to an application heartbeat test for connectivity. Although the default time interval is two hours, reducing it to something more useful to an application (for example, 5 minutes) may create performance and network load issues and should be done only after extensive analysis.</p> <p>This variable is also <code>keepalive_time</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 4 of 16)**

Variable	Description or Description Location
tcpVosKeepaliveKeepaliveTries	<p>The number of keepalive probes sent before the connection is declared failed.</p> <p>This variable is also <code>keepalive_tries</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosKeepaliveCheckIfDead	<p>A timer for the succeeding probes.</p> <p>This variable is also <code>check_if_dead</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosMaxRetryInterval	<p>The maximum retransmission time for unacknowledged segments. Values are the character string <code>2*tcp_msl</code> (the default) or an integer in the range 15 to 240. <code>tcp_msl</code> is the value of the maximum segment lifetime parameter. The <code>max_retry_interval</code> parameter is not an abort timer; rather, <code>max_retry_interval</code> sets an upper limit on a backoff timer, which is the amount of time TCP waits before retransmitting a segment. The backoff timer starts at a value computed from the round-trip time, but then increases its value as the number of retransmissions increases, in order to avoid swamping the network.</p> <p>The <code>max_retry_interval</code> parameter is different from intervals set by <code>estab_abort</code>, <code>syn_sent_abort</code>, and <code>syn_rcvd_abort</code>. The <code>max_retry_interval</code> parameter sets the maximum retransmission time for packets, while the three abort timers set a limit on the total time before aborting. If the abort timers are not set (the default), the <code>max_retry_interval</code> parameter indirectly influences the abort time in that connections are aborted after some number of unacknowledged retransmissions.</p> <p>This variable is also <code>max_retry_interval</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 5 of 16)**

Variable	Description or Description Location
tcpVosEstabAbortInterval	<p>The time, in seconds, to abort an unacknowledged connection when TCP is in the <code>ESTAB</code> state. Set this parameter when you do not wish to wait for the standard timeout. The standard timeout depends on the retransmission timer (which depends on the measured round-trip time, a backoff adjustment made for each retry, and multiple retries). A typical timeout is between eight to ten minutes. By default, this parameter is set to <code>off</code>. This parameter does not change the algorithm or timers used to retransmit unacknowledged packets; however, setting this parameter results in fewer retransmissions before the connection is aborted because it establishes an absolute interval from the first transmit. Setting the parameter value too small may cause a recoverable connection to be terminated. This parameter is different from setting the <code>max retry interval</code>, which limits the maximum retransmission time for packets, not the total time. This parameter limits the total time.</p> <p>This variable is also <code>estab_abort</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosSynsentAbortInterval	<p>The time, in seconds, to abort an unacknowledged connection when TCP is in the <code>SYN_SENT</code> state during an application's open connect process. A socket enters the <code>SYN_SENT</code> state when TCP sends a connection request but has not yet received an acknowledgment for the connection request. The standard timeout depends on the <i>retransmission timer</i>, which is a backoff adjustment made for each retry and multiple retries. A typical timeout is about two and one-half minutes. By default, this parameter is set to <code>off</code>. This parameter does not change the algorithm or timers used to retransmit unacknowledged packets; however, setting this parameter results in fewer retransmissions before the connection is aborted because it establishes an absolute interval from the first transmit. Setting the parameter value too small may prevent the socket from establishing connections with valid clients. This parameter is different from setting the <code>max retry interval</code>, which limits the maximum retransmission time for packets, not the total time. This parameter limits the total time.</p> <p>This variable is also <code>syn_sent_abort</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 6 of 16)**

Variable	Description or Description Location
tcpVosSynrcvdAbortInterval	<p>The time, in seconds, to abort an unacknowledged connection when TCP is in the <code>SYN_RCVD</code> state during an application's open connect process. A socket enters the <code>SYN_RCVD</code> state when TCP receives a connection request, sends the answering packet, but has not yet received an acknowledgment for the answering packet. A typical timeout is about two and one-half minutes. By default, this parameter is set to <code>off</code>. This parameter does not change the algorithm or timers used to retransmit unacknowledged packets; however, setting this parameter results in fewer retransmissions before the connection is aborted because it establishes an absolute interval from the first transmit. Setting the parameter to an interval value reduces your vulnerability to various denial-of-service attacks where connection requests are received for hosts that do not exist. Setting the parameter value too small may prevent the socket from establishing connections with valid clients. This parameter is different from setting the <code>max_retry_interval</code>, which limits the maximum retransmission time for packets, not the total time. This parameter limits the total time.</p> <p>This variable is also <code>syn_rcvd_abort</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosEnableMetering	<p>A switch that enables TCP metering, both global and on a per TCP transmission control block (TCB) basis. Any TCBs created when this is set to <code>off</code> never contain metering information, even when this parameter is turned on. For TCBs created when this is <code>on</code>, metering information accumulates only when metering is enabled.</p> <p>This variable is also <code>tcp_metering</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosSendLegacyMSS	<p>A switch that controls what TCP sends for the MSS on the initial SYN packets. The <code>off</code> value (0) generates the RFC correct value, which is derived from the MTU for the local subnet. The <code>on</code> value (1) generates an MSS value that takes into account routing.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 7 of 16)**

Variable	Description or Description Location
tcpVosMaxSend tcpVosUseBigWindows tcpVosMaxSmallWindow tcpVosCurNum8kWindows tcpVosCurNum16kWindows tcpVosCurNum32kWindows tcpVosCurNum64kWindows tcpVosCurNum256kWindows tcpVosCurNumHugeWindows tcpVosMaxNum16kWindows tcpVosMaxNum32kWindows tcpVosMaxNum64kWindows tcpVosMaxNum256kWindows tcpVosMaxNumHugeWindows	<p>Current values for STCP window sizes, system wide (for information, see the <i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420)). These variables are also in <code>list_stcp_params</code> and <code>set_stcp_param</code>, as follows:</p> <p> max_send_ws  big_windows  dft_recv_ws  current 8k windows  current 16k windows  current 32k windows  current 64k windows  current 256k windows  current huge windows  max_16k_windows  max_32k_windows  max_64k_windows in  max_256k_windows  max_huge_windows </p>
tcpVosDetectSynAttack	<p>A switch that allows STCP to optimize its ability to respond to DOS attacks. When the value of <code>detect_syn_attack</code> is on, STCP ignores requests from a client that sends an unreasonable number of failed connect requests over an extended time period, until the client reduces its rate of requests. The TCP protocol requires that a host respond to all connect requests by sending a reset (RST) reply packet to indicate that the requests have been denied. In legitimate situations, a host typically sends RST packets if it is not listening to the target port or if it has reached its listen backlog limit. Setting the value of <code>detect_syn_attack</code> to on temporarily overrides STCP's typical behavior for specific targeted sites, and enables STCP to log such activity and to identify the source of the packets. By default, the value of <code>detect_syn_attack</code> is off. See <a href="#">Chapter 7</a> for information about using this parameter.</p> <p>This variable is also <code>detect_syn_attack</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 8 of 16)**

Variable	Description or Description Location
tcpVosPreventPortScanning	<p>A switch that allows STCP to prevent port scanning. In <i>port scanning</i>, a client sends connect requests (that is, SYN packets) to all ports on a host in order to gain information about that host. The TCP protocol requires a host to send an RST packet when it receives a connect request at a port that it is not listening to. When a client receives an RST packet, the client learns that the host is reachable and also that the host is not listening to that port.</p> <p>You can enable STCP to defend itself against port scanning by using the <code>prevent_port_scan</code> parameter. When <code>prevent_port_scan</code> is on, STCP does <b>not</b> send an RST packet when it receives a connect request at a port that it is not listening to. As a result, a user has a more difficult time obtaining this information from a Stratus module and using the information for malicious activity. However, when <code>prevent_port_scan</code> is on, STCP behavior becomes non-standard, and legitimate debugging activities may be more difficult. See <a href="#">Chapter 7</a> for information about using this parameter.</p> <p>This variable is also <code>prevent_port_scan</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>
tcpVosDelayedAckInterval	<p>The amount of time that TCP delays sending an acknowledgement in order to combine with a data message, if any, that is replying to the message that needs an acknowledgement. The typical value is 200. Do not specify a value greater than 200.</p> <p>This variable is also <code>ack_delay</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 9 of 16)**

Variable	Description or Description Location
tcpVosBadSeqResetLimit tcpVosBadAckResetLimit	<p>These fields enable STCP to reset a connection when the number of bad segments received with no intervening good segments exceeds the value specified by the relevant parameter. Bad sequence numbers and bad ACK numbers are values that are impossible given the current connection state. More precisely, they are sequence and ACK numbers that fail the segment validation rules in RFC-793 as amended by RFC-1122. Values are the character string <code>off</code> (the default) or non-zero positive integers.</p> <p>STCP keeps a counter for the number of segments with bad sequence numbers and another counter for the number of bad acknowledgements. STCP increments the counters each time bad data is received and decrements the counters when good data is received. If a counter exceeds the value specified by the appropriate parameter, <code>bad_seq_reset_limit</code> or <code>bad_ack_reset_limit</code>, STCP resets the connection instead of sending an ACK. These variables are also in <code>list_stcp_params</code> and <code>set_stcp_param</code>, as follows:</p> <p><code>bad_seq_reset_limit</code>  <code>bad_ack_reset_limit</code></p>
tcpVosUseRfc1122Urgent	<p>A switch that turns on or off the RFC-1122 implementation for processing urgent data. The switch is captured when a socket is created and may be altered afterward using the socket-level option <code>_TCP_STDURG</code>. This variable is also <code>rfc_1122_urgent</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code></p>
tcpVosReuseAddrAction	<p>A switch that controls the functionality of the <code>SO_REUSEADDR</code> option of the <code>setsockopt</code> function. For information, see <i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420). This variable is also <code>tcp_reuseaddr_action</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code>.</p>

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 10 of 16)**

Variable	Description or Description Location
tcpVosRfc1323Config	Whether or not STCP attempts window scaling, Round-Trip Time Measurement (RTTM), and Protect Against Wrapped Sequence Numbers (PAWS). Whenever possible, use RTTM and PAWS on links with a speed of 1Gb or greater, to ensure the best data integrity. Window scaling and Selective Acknowledgement (SACK) are strictly performance enhancements. Values are: <ul style="list-style-type: none"> <li>– <code>disable</code>—STCP does not attempt to negotiate for window scaling, RTTM, and PAWS, and it rejects requests from the peer to use them.</li> <li>– <code>allow</code>—STCP does not attempt to negotiate for these features but allows the peer to request them.</li> <li>– <code>request</code>—STCP attempts to negotiate for these features and allows the peer to request them.</li> </ul> This variable is also <code>tcp_rfc_1323_policy</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code> .
tcpVosWindowScale	The value STCP uses when attempting to negotiate for window scaling. The value 0 turns off window scaling but allows RTTM and PAWS to remain enabled. This variable is also <code>tcp_window_scale</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code> .
tcpVosSackConfig	Whether or not STCP attempts SACK. Values are: <ul style="list-style-type: none"> <li>– <code>disable</code>—STCP does not attempt SACK, and it rejects requests from the peer to use it.</li> <li>– <code>allow</code>—STCP does not attempt SACK but allows the peer to request it.</li> <li>– <code>request</code>—STCP attempts SACK and allows the peer to request it.</li> </ul> This variable is also <code>tcp_SACK_policy</code> in <code>list_stcp_params</code> and <code>set_stcp_param</code> .
tcpVosDebug	Stratus internal use only.
tcpVosDropAck	
tcpVosAllowUnwiseRFC1323Options	The number of peers allowed to connect with unwise combinations of RFC-1323 options.
tcpVosDiscardSynfin	The number of discard packets containing both SYN and FIN.
tcpVosZerowinAbortInterval	The amount of time to allow a zero window from the peer (zero means forever).



**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 11 of 16)**

Variable	Description or Description Location
tcpVosMaxBacklog	The maximum number of unaccepted incoming connections allowed (per listening socket). This is the upper limit applied to the backlog count given to the listen call.
tcpVosMaxBacklogCache	The maximum number of TCBs allocated and ready to handle incoming connections (per listening socket).
tcpVosBytesReceivedInOrder	The number of bytes fast mode delivered when all data arrives in correct sequence and no buffering is required due to flow control or the existence of previous segments with a lower sequence number. The segment passes immediately upstream. This variable is also using fast mode in stcp_meters.
tcpVosFlowControlledWhenReceived	The number of segments received when the application or an upper streams module exerts flow control. This variable is also flow controlled when received in stcp_meters.
tcpVosDataPendingWhenReceived	The number of segments received when other buffered data is waiting to be passed upstream. This variable is also data pending when received in stcp_meters.
tcpVosReceivedOutOfOrder	The number of segments received that are inside the receive window but that do not have the expected sequence number. These segments are buffered. This variable is also received out of order in stcp_meters.
tcpVosSeqnoBeforeExpected	The number of segments received that are inside the receive window but that do not have the expected sequence number. These segments are buffered. This variable is also seqno before expected in stcp_meters.
tcpVosSeqnoAfterExpected	The number of segments received that are inside the receive window but that do not have the expected sequence number. These segments are buffered. This variable is also seqno after expected in stcp_meters.
tcpVosSegmentOutsideWindow	The number of segments received that extend outside the receive window. These segments are truncated. This variable is also segment outside window in stcp_meters.

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 12 of 16)**

Variable	Description or Description Location
tcpVosSegmentsInOrder	The number of segments received in fast mode. This corresponds with tcpVosBytesReceivedInOrder (the fast mode byte count). This variable is also fast path in stcp_meters.
tcpVosUrgentDataInOrder	The number of urgent data packets processed in fast mode. This variable is also fast path:urgent data in stcp_meters.
tcpVosUrgentDataOutOfOrder	The number of urgent data packets processed in slow mode. This variable is also slow pathurgent data in stcp_meters.
tcpVosOverlapOccurred	The number of segment overlaps involving those due to handling of urgent data in slow mode. This variable is also overlap occurred in stcp_meters.
tcpVosBacklapOccurred	The number of urgent data packets processed in slow mode. This variable is also backlap occurred in stcp_meters.
tcpVosWindowClosingWhenReceived	The number of segments received when the advertised receive window size is reduced in size. This variable is also window < max when received in stcp_meters.
tcpVosWindowSetZeroAfterReceive	The number of segments received that cause the receive window size to be reduced to zero. This variable is also window set zero after receive in stcp_meters.
tcpVosProbesSent	The number of probes sent. When the send window size becomes zero (preventing further transmissions), a probe is sent asking for a re-acknowledgement of a byte that was already sent. This variable is also probes sent in stcp_meters.
tcpVosKeep_alivesSent	The number of keepalive probes sent. TCP sends a keepalive probe if the keepalive socket option is set for a TCP socket and no data is exchanged for a period of time. The default time is two hours. This variable is also keep_alives sent in stcp_meters.

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 13 of 16)**

Variable	Description or Description Location
tcpVosExplicitAcksSent	Count of one of many conditions under which an ACK must be sent without waiting for a segment on which to piggy-back. (Usually, ACKs attach to segments containing other information; that is, they are piggy-backed. In some cases, ACKs are sent even if there is no other data to send.) This variable is also <code>explicit acks sent</code> in <code>stcp_meters</code> .
tcpVosAcksWithMutlipleSegUnacked	The number of segments acknowledged immediately because more than one full size segment was unacknowledged ( <code>unacked</code> ). This variable is also due to <code>&gt; 1 segment unacked</code> in <code>stcp_meters</code> .
tcpVosAcksDueToWindowSizeIncrease	The number of segments acknowledged immediately because the window size increased by more than 60% to the maximum size. This variable is also due to <code>window size increase</code> in <code>stcp_meters</code> .
tcpVosAcksDueToNonzeroWindowSize	The number of segments acknowledged immediately because the window size went from zero to nonzero. This variable is also due to <code>nonzero window size</code> in <code>stcp_meters</code> .
tcpVosDuplicateAcksSent	The sum of <code>explicit acks</code> and <code>timer acks</code> that have a duplicate sequence number already acknowledged. This variable is also <code>duplicate acks sent</code> in <code>stcp_meters</code> .
tcpVosReadServiceRtnExecuted	The number of times the <code>read service routine</code> is called when buffered data is pending. This variable is also <code>read service rtn executed</code> in <code>stcp_meters</code> .
tcpVosDataPending	The number of times the <code>read service routine</code> is called when buffered data is pending. This variable is also <code>data pending</code> in <code>stcp_meters</code> .
tcpVosDataPendingBeyondSeqno	The number of times all buffered data is beyond the expected sequence number. In this case, no data can pass upstream until the missing segment is received. If the missing segment is not received in a specified time and, therefore, is not acknowledged, the sender retransmits the segment. This variable is also <code>all pending beyond seqno</code> in <code>stcp_meters</code> .
tcpVosQueueEmpty	The number of times the <code>read service routine</code> is called with an empty queue. This variable is also <code>queue empty</code> in <code>stcp_meters</code> .

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 14 of 16)**

Variable	Description or Description Location
tcpVosFlowCtlBeforeProcessingQ	The number of times the read service routine is called when no segments can pass upstream due to flow control exerted at the stream head. This variable is also flow ctl before processing q in stcp_meters.
tcpVosFlowCtlAfterProcessingQ	The number of times the read service routine is called when flow control is exerted during the process of passing messages upstream. This variable is also flow ctl after processing q in stcp_meters.
tcpVosBufdatExecuted	The number of times the bufdat routine is called to unbuffer segments. It is called by the read service routine when all of the following conditions are true: flow control is not exerted; the left edge of the receive window is advancing; and the connection is not in fast mode (fast mode transmission implies no data buffering is pending). This variable is also bufdat executed in stcp_meters.
tcpVosBufdatWithWindowSetToMax	The number of times the bufdat routine processing has unbuffered all segments in the window and the advertised receive window size is reset to the maximum size. This variable is also window set to max in stcp_meters.
tcpVosAllocationFailures	The number of allocation failures. This variable is also allocation failures in stcp_meters.
tcpVosUnknownPeerResets	The number of resets sent when no socket was found.
tcpVosInvalidAckResets	The number of resets sent because the ACK number was invalid.
tcpVosInvalidSynAckResets	The number of resets sent because the ACK number The number was invalid in SYN-SENT state.
tcpVosAckingUnsentDataResets	The number of resets sent because the ACK number was for unsent data.
tcpVosInvalidSynResets	The number of resets sent because an invalid SYN was received by a listening socket.
tcpVosAckOnFirstSynResets	The number of resets sent because a SYN with an ACK was received by a listening socket.
tcpVosDataAfterCloseResets	The number of resets sent because data was received after the socket was unable to receive data.

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext (Page 15 of 16)**

Variable	Description or Description Location
tcpVosCloseDropResets	The number of resets sent because an application closed a socket when more input data was pending.
tcpVosTDisconResets	The number of resets sent because a listening application issued a TLI T_DISCON_REQ for one of the sockets in the backlog queue.
tcpVosResetsInBacklog	The number of resets received from the peer while the socket was still in the backlog queue.
tcpVosListenerCloseResets	The number of resets sent for sockets in the backlog queue when the listening socket is closed.
tcpVosLingerResets	The number of resets sent because an application specified linger with a value of zero.
tcpVosDuplicatePeerResets	The number of resets sent because a new SYN was received for a combination of IP addresses and ports that is already in use (bug on the peer machine).
tcpVosSynBadStateResets	The number of resets sent because a SYN was received in the wrong state.
tcpVosSnmpResets	The number of resets sent because an SNMP request was used to reset the connection.
tcpVosStackDownResets	The number of resets sent because someone brought the TCP stack down.
tcpVosTDisconSelfResets	The number of resets sent because an application sent a TLI T_DISCON_REQ message with no connection specified (i.e., aborted the connection the T_DISCON_REQ was sent on).
tcpVosBadSeqResets	The number of resets sent because the number of consecutive segments with bad sequence numbers exceeded the threshold.
tcpVosBadAckResets	The number of resets sent because the number of consecutive segments with bad ACK numbers exceeded the threshold.
tcpVosProbeTimeoutResets	The number of resets sent because the number unacknowledged probes exceeded the limit.
tcpVosKeepaliveTimeoutResets	The number of resets sent because the number unacknowledged keepalives exceeded the limit.

**Table 9-14. Fields Displayed by netstat -protocol tcp/ext** (Page 16 of 16)

Variable	Description or Description Location
tcpVosSynRtxTimeoutResets	The number of resets sent because the number of retransmissions for SYNs was exceeded.
tcpVosRtxTimeoutResets	The number of resets sent because the number of retransmissions for data was exceeded.
tcpVosFinRtxTimeoutResets	The number of resets sent because the number of retransmissions for FINs was exceeded.
tcpVosSynSentAbortTimeoutResets	The number of resets sent because the Syn Sent abort timeout was exceeded.
tcpVosSynRcvdAbortTimeoutResets	The number of resets sent because the Syn Received abort timeout was exceeded.
tcpVosEstabAbortTimeoutResets	The number of resets sent because the established abort timeout was exceeded.
tcpVosFinalTimeoutResets	The number of resets sent because the final timeout was exceeded.
tcpVosZerowinAbortTimeoutResets	The number of resets sent because the zero window abort criteria had been exceeded.
tcpVosLingerDataResets	The number of resets sent because input data was present when a LINGER was begun.
tcpVosLingerTimeoutResets	The number of resets sent if data didn't drain out before linger interval completed.
tcpVosShutdownDataResets	The number of resets sent if input data was discarded by a write shutdown.

[Table 9-15](#) describes the fields that report UDP statistics. The fields are described in the order that they appear in the output.

**Table 9-15. Fields in netstat That Report UDP Statistics** (Page 1 of 2)

Field	Description
udpInDatagrams	The number of UDP datagrams delivered to users (those datagrams received without errors, and with the application listening at the correct port).
udpNoPorts	The count of received nonbroadcast datagrams for a port to which no one is listening (an unbound port). A “destination unreachable” message is generated. This may indicate a local application failure or possibly a client/server coordination or design problem.

**Table 9-15. Fields in netstat That Report UDP Statistics** (Page 2 of 2)

Field	Description
udpInErrors	The number of UDP datagrams that could not be delivered due to errors (the sum of incomplete headers, bad data length fields, bad checksums, and a broadcast packet on a port to which no one is listening). A sudden change in rate may indicate that there is a misconfigured host, routers that need to update their routes more frequently, or a problem with the application.
udpOutDatagrams	The number of UDP datagrams sent (even failures are counted).
udpVosCksumErrors	The number of UDP datagrams with checksum errors. This may indicate network corruption.
udpVosSmallHeaders	The number of UDP datagrams with small headers. This may indicate network corruption.
udpVosTruncatedDatagrams	The number of UDP datagrams with a bad length field value. This may indicate network corruption.
udpVosHCInBytes	The number of UDP data bytes received.
udpVosHCOutBytes	The number of UDP data bytes sent.

[Table 9-16](#) describes some of the fields that report AF UNIX statistics. For additional fields, see the `VOSAFLOCAL.MIB.mib` file in `(master_disk)>system>vos_mibs`.

**Table 9-16. Fields in netstat That Report AF UNIX Statistics** (Page 1 of 2)

Field	Description
<b>Global statistics</b>	
current num sockets	The current number of open sockets.
current num connections	The current number of connections between sockets, which is half the number of connected sockets.
binds	The number of times a socket was bound to an actual address.
sockets created	The number of times a socket was opened.
sockets closed	The number of times a socket was closed.
shutdowns	The number of successful shutdown calls.
orderly disconnects	The number of connections that were closed without data loss.
connection resets	The number of connections that were closed abnormally; that is, the connections were reset.

**Table 9-16. Fields in netstat That Report AF UNIX Statistics** (Page 2 of 2)

Field	Description
<b>Rejected connection statistics</b>	
by user	The number of incoming connect requests rejected by the <code>accept</code> runtime code. Connections are rejected when the runtime cannot create a new socket on which to accept the connection. The most likely cause is a clone limit that is set too low.
backlog full	The number of connect requests that were rejected because the backlog of the specified peer was full; that is, the connect requests were rejected because the number of connections pending exceeds the number specified in the peer's listen request.
peer not listening	The number of connect requests that were rejected because the specified peer was not listening for incoming connections.
peer not present	The number of connect requests that were rejected because the specified peer did not exist.
<b>Discarded message statistics</b>	
no peer	The number of <code>SOCK_DGRAM</code> messages dropped because the specified peer did not exist or was in the wrong state to receive messages.
outstate	The number of messages dropped because the sender was in the wrong state.

The statistics may indicate certain types of problems. [Table 9-17](#) lists some problems and the fields that may report them.



**Table 9-17. Problems Indicated by Fields in the netstat Statistics** (Page 1 of 2)

Problem	Field
Data-link corruption or transmission errors at the source or intermediate router	<p>In the TCP section:</p> <ul style="list-style-type: none"> <li>- tcpVosHeaderTruncated</li> <li>- tcpRetransSegs</li> <li>- tcpInErrs</li> </ul> <p>In the UDP section:</p> <ul style="list-style-type: none"> <li>- udpVosSmallHeaders</li> <li>- udpVosChecksumErrors</li> <li>- udpVosTruncatedDatagrams</li> <li>- udpInErrors</li> </ul> <p>In the IP section:</p> <ul style="list-style-type: none"> <li>- ipVosSystemStatsInSmallHeader</li> <li>- ipVosSystemStatsInChecksumErr</li> <li>- ipVosSystemStatsInTruncatedPkt</li> <li>- ipVosSystemStatsInTooBig</li> </ul> <p>- ipInHdrErrors (which includes the previous errors as well as the Time-To-Live exceeded count)</p> <p>In the ICMP section:</p> <ul style="list-style-type: none"> <li>- icmpInErrors</li> </ul>
Indication of routing errors	<p>In the IP section:</p> <ul style="list-style-type: none"> <li>- ipInAddrErrors</li> <li>- ipForwDatagrams</li> <li>- ipInUnknownProtos</li> <li>- ipOutNoRoutes</li> </ul> <p>In the ICMP section:</p> <ul style="list-style-type: none"> <li>- icmpInDestUnreaches</li> <li>- icmpInTimeExcds</li> </ul>
Local host memory problems	<p>In the IP section:</p> <ul style="list-style-type: none"> <li>- ipInDiscards</li> <li>- ipRoutingDiscards</li> <li>- ipOutDiscards</li> </ul> <p>In the ICMP section:</p> <ul style="list-style-type: none"> <li>- icmpOutErrors</li> <li>- icmpOutSrcQuenches</li> </ul>

**Table 9-17. Problems Indicated by Fields in the netstat Statistics** (Page 2 of 2)

Problem	Field
Remote host memory problems	In the ICMP section: – icmpInSrcQuenchs
Application problems	In the UDP section: – datagrams received, no application for port requested  In the ICMP section: – icmpInDestUnreachs – icmpOutDestUnreachs

## Sample Output of the netstat Command

The following sample output of the `netstat` command shows the default display when TCP/IP connections are active.

### netstat

#### Connected TCP Endpoints

Recv	Recv Q	Sent	Send Q	Local Address	Foreign Address	State
93.8K	0	14.1M	0	10.0.1.110:3002	10.0.1.3:62705	established
16.2K	0	14.3K	0	10.0.1.110:3002	10.0.1.108:49748	established
743K	0	108M	0	10.0.1.110:3005	10.0.1.3:62341	established
681K	0	28.3K	0	10.0.1.110:49637	10.0.1.121:3000	established
595	0	293	0	10.0.2.110:3001	10.0.2.121:49363	established
63.1K	0	102K	0	10.0.2.110:49167	10.0.2.3:3000	established
9.84K	0	32.0K	52	swslem110:ssh	ROBINSON-H:61577	established
729	0	33.4K	0	swslem110:telnet	KO-H1:55389	established
895	0	101K	0	swslem110:telnet	KO-H1:55392	established

#### Active UDP Endpoints

Recv	Bytes	Send	Bytes	Con #	Process	Local Address	Foreign Ad
				2	071C804	*:discard	*:*
				1	071C804	*:daytime	*:*
				3	071C804	*:chargen	*:*
				4	071C804	*:time	*:*
				5	071C804	*:tftp	*:*
				15	071C809	*:ntp	*:*
512	26.9K			6	071C806	*:nb_nmsrv	*:*
306	65.7K			7	071C806	*:nb_dgsrv	*:*
				26	071C809	10.0.1.110:ntp	*:*
				28	071C809	10.0.2.110:ntp	*:*
				33	071C809	10.0.244.110:ntp	*:*
18	1.11K	134	7.78K	8	071C806	10.0.244.110:nb_nmsrv	*:*

(Continued on next page)

		31	6.79K	9	071C806	10.0.244.110:nb_dgsrv	*:*
				36	071C809	10.2.110.110:ntp	*:*
				40	071C809	10.2.110.210:ntp	*:*
17	816	17	816	22	071C809	10.10.1.1:ntp	*:*
				30	071C809	10.224.0.110:ntp	*:*
		60	4.08K	10	071C806	10.224.0.110:nb_nmsrv	*:*
		31	6.79K	11	071C806	10.224.0.110:nb_dgsrv	*:*
				20	071C809	localhost.stratus.com:ntp	*:*
84	4.03K	84	4.03K	24	071C809	swslem110:ntp	*:*
49	2.75K	69	4.83K	12	071C806	swslem110:nb_nmsrv	*:*
		13	3.13K	13	071C806	swslem110:nb_dgsrv	*:*
				42	071C809	204.252.133.100:ntp	*:*
				45	071C809	204.252.134.101:ntp	*:*
				16	071C809	*:ntp	*:*
				14	071C80A	*:31002	*:*
				43	071C809	2001:1900:3008:fc85::64:nt	*:*
				46	071C809	2001:1900:3008:fc86::65:nt	*:*
				34	071C809	fd73:738c:286:f4::6e:ntp	*:*
				37	071C809	fd73:738c:286:26e::6e:ntp	*:*
				38	071C809	fd73:738c:286:26e:200:a8ff	*:*
				31	071C809	fd73:738c:286:e000::6e:ntp	*:*
				19	071C809	fe80::1%1:ntp	*:*
				21	071C809	fe80::200:a8ff:fe40:3385%2	*:*
				23	071C809	fe80::200:a8ff:fe41:3385%3	*:*
				25	071C809	fe80::200:a8ff:fe42:3385%4	*:*
				27	071C809	fe80::200:a8ff:fe43:3385%5	*:*
				29	071C809	fe80::200:a8ff:fe44:3385%6	*:*
				32	071C809	fe80::200:a8ff:fe45:3385%7	*:*
				35	071C809	fe80::200:a8ff:fe46:3385%8	*:*
				39	071C809	fe80::200:a8ff:fe47:3385%9	*:*
				41	071C809	fe80::200:a8ff:fe49:3385%1	*:*
				44	071C809	fe80::200:a8ff:fe4a:3385%1	*:*

netstat -statistics

tcp

vanj		tcpRtoAlgorithm
320		tcpRtoMin
500K		tcpRtoMax
-1		tcpMaxConn
70.0K		tcpVosMaxTCBs
1.02K		tcpVosFirstNonPrivilegedPort
49.1K		tcpVosFirstDynamicPort
65.5K		tcpVosFirstSharedDynamicPort
394		tcpActiveOpens
37		tcpPassiveOpens
9		tcpCurrEstab
101K		tcpInSegs
122K		tcpOutSegs

(Continued on next page)

397		tcpOutRsts
50.1K		tcpVosDataSegmentsReceived
30.6M		tcpVosBytesReceived
98.8K		tcpVosSegmentsSent
123M		tcpVosTotalBytesSent
327		tcpVosSocketsAllocated

## udp

1.42K		udpInDatagrams
271		udpNoPorts
608		udpOutDatagrams
45.7K		udpVosHCOutBytes

## af\_local (af\_unix)

2		vosAfLocalCurrentNumSockets
2		vosAfLocalBinds
2		vosAfLocalSocketsCreated

## ip

ipv4		ipv6	
0:01:05.041		0:01:05.041	ipSystemStatsDiscontinuityTime
1.00K		1.00K	ipSystemStatsRefreshRate
111K		59	ipSystemStatsInReceives
36.9M		4.86K	ipSystemStatsInOctets
21			ipSystemStatsInAddrErrors
20			ipSystemStatsInUnknownProtos
41			ipSystemStatsInDiscards
111K		59	ipSystemStatsInDelivers
124K		139	ipSystemStatsOutRequests
124K		138	ipSystemStatsOutTransmits
130M		9.56K	ipSystemStatsOutOctets
8.54K		51	ipSystemStatsInMcastPkts
798K		4.32K	ipSystemStatsInMcastOctets
1.25K		54	ipSystemStatsOutMcastPkts
113K		3.54K	ipSystemStatsOutMcastOctets
913			ipSystemStatsInBcastPkts
177			ipSystemStatsOutBcastPkts
33			ipVosSystemStatsRtsUnreach
		9	ipVosSystemStatsExtHdrTooLongs

## arp

600		arpVosCacheLife
60		arpVosRequestTimeout
16		arpVosMaxQueuedDatagrams

(Continued on next page)

```

70.0K | arpVosGratuitousLimit
55    | arpVosArpTableEntries
icmp

```

ipv4	ipv6
	59 icmpStatsInMsgs
	139 icmpStatsOutMsgs

In Packets	Out Packets	Type
7		v6Echos
	7	v6EchoReplies
	9	v6NdRouterSolicits
27		v6NdRouterAdverts
24	71	v6NdNeighborSolicits
1	24	v6NdNeighborAdverts
	28	v6Mldv2ListenerReports

42		vosIgmpRcvTotal
42		vosIgmpRcvTooshort

8.52K	vosRawHCInReceives
631K	vosRawHCInOctets
1.28K	vosRawHCOutRequests
91.6K	vosRawHCOutOctets

%swsle#loop.m110	ifName
online	ifStackStatus
localhost.stratus.com/8	ipAddressPrefixPrefix
localhost	ipAddressPrefixPrefix
fe80::1%1/64	ipAddressPrefixPrefix
224.0.0.1	mgmdInverseHostCacheAddress
ff01::1	mgmdInverseHostCacheAddress
ff02::1	mgmdInverseHostCacheAddress
ff02::1:ff00:1	mgmdInverseHostCacheAddress
1	ifIndex
Internal Loopback	ifDescr
16.3K	ifMtu
000000 000000	ifPhysAddress
up	ifOperStatus
0:01:05.086	ifLastChange

```

0:01:05.086 | ifCounterDiscontinuityTime
0           | ifHighSpeed

```

%swsle#sdlmux.10.2.0	#enet.10.2.0	#enet.11.2.0	ifName
online	online	standby	ifStackStatus
10.10.1.1/24			ipAddressPrefixPref
fe80::200:a8ff:fe40:			ipAddressPrefixPref
224.0.0.1			mgmdInverseHostCach
ff01::1			mgmdInverseHostCach
ff02::1			mgmdInverseHostCach
ff02::1:ff40:3385			mgmdInverseHostCach
2	3	4	ifIndex
Link Multiplexor	80003ES2LAN Giga	80003ES2LAN Giga	ifDescr
9.20K	9.20K	9.20K	ifMtu
0000A8 403385	0000A8 403385	0000A8 603385	ifPhysAddress
up	up	up	ifOperStatus
0:03:02.098	0:01:57.068	0:01:59.065	ifLastChange
0:01:06.005	0:01:57.068	0:01:59.065	ifCounterDiscontinui
	fullDuplex	fullDuplex	dot3StatsDuplexStat
0	100M	100M	ifHighSpeed
	32.2M	58.1K	ifInOctets
	45.5K	588	ifInUcastPkts
	1.97M	37.7K	ifOutOctets
	27.5K	590	ifOutUcastPkts
	167	200	ifInBroadcastPkts
	3		ifOutMulticastPkts
	33		ifOutBroadcastPkts

%swsle#sdlmux.10.8.0	#enet.10.8.0	#enet.11.8.0	ifName
online	online	standby	ifStackStatus
swslem110/24			ipAddressPrefixPrefi
fe80::200:a8ff:fe41:3			ipAddressPrefixPrefi
224.0.0.1			mgmdInverseHostCache
ff01::1			mgmdInverseHostCache
ff02::1			mgmdInverseHostCache
ff02::1:ff41:3385			mgmdInverseHostCache
5	6	7	ifIndex
Link Multiplexor	82546 R3 Gigabi	82546 R3 Gigabi	ifDescr
9.20K	9.20K	9.20K	ifMtu
0000A8 413385	0000A8 413385	0000A8 613385	ifPhysAddress
up	up	up	ifOperStatus
0:03:03.013	0:02:09.099	0:02:16.090	ifLastChange
0:02:01.083	0:02:03.087	0:02:10.066	ifCounterDiscontinui
	fullDuplex	fullDuplex	dot3StatsDuplexStatu
0	1.00G	1.00G	ifHighSpeed
	642K	234K	ifInOctets
	5.34K	589	ifInUcastPkts
	642K	37.7K	ifOutOctets
	5.71K	590	ifOutUcastPkts
	42		ifInMulticastPkts

(Continued on next page)

	2.01K	2.07K	ifInBroadcastPkts
	7		ifOutMulticastPkts
	53		ifOutBroadcastPkts
%swsle#sdlmux.10.8.1	%swsle#enet.10.8.1	ifName	
online	online	ifStackStatus	
10.0.1.110/24		ipAddressPrefixPrefix	
fe80::200:a8ff:fe42:3385%4/64		ipAddressPrefixPrefix	
224.0.0.1		mgmdInverseHostCacheAddress	
ff01::1		mgmdInverseHostCacheAddress	
ff02::1		mgmdInverseHostCacheAddress	
ff02::1:ff42:3385		mgmdInverseHostCacheAddress	
8	9	ifIndex	
Link Multiplexor	82546 R3 Gigabit NI	ifDescr	
9.20K	9.20K	ifMtu	
0000A8 423385	0000A8 423385	ifPhysAddress	
up	up	ifOperStatus	
0:03:03.031	0:02:15.078	ifLastChange	
0:02:15.072	0:02:15.078	ifCounterDiscontinuityTime	
	fullDuplex	dot3StatsDuplexStatus	
0	1.00G	ifHighSpeed	
	5.24M	ifInOctets	
	52.7K	ifInUcastPkts	
	129M	ifOutOctets	
	91.6K	ifOutUcastPkts	
	281	ifInBroadcastPkts	
	3	ifOutMulticastPkts	
	4	ifOutBroadcastPkts	
%swsle#sdlmux.11.8.1	%swsle#enet.11.8.1	ifName	
online	online	ifStackStatus	
10.0.2.110/24		ipAddressPrefixPrefix	
fe80::200:a8ff:fe43:3385%5/64		ipAddressPrefixPrefix	
224.0.0.1		mgmdInverseHostCacheAddress	
ff01::1		mgmdInverseHostCacheAddress	
ff02::1		mgmdInverseHostCacheAddress	
ff02::1:ff43:3385		mgmdInverseHostCacheAddress	
10	11	ifIndex	
Link Multiplexor	82546 R3 Gigabit NI	ifDescr	
9.20K	9.20K	ifMtu	
0000A8 433385	0000A8 433385	ifPhysAddress	
up	up	ifOperStatus	
0:03:03.049	0:02:18.076	ifLastChange	
0:02:18.070	0:02:18.076	ifCounterDiscontinuityTime	
	fullDuplex	dot3StatsDuplexStatus	
0	1.00G	ifHighSpeed	
	107K	ifInOctets	
	379	ifInUcastPkts	
	131K	ifOutOctets	
	407	ifOutUcastPkts	

(Continued on next page)

	270	ifInBroadcastPkts
	3	ifOutMulticastPkts
	1	ifOutBroadcastPkts

%swsle#sdlmux.10.8.2	#enet.10.8.2	#enet.11.8.2	ifName
online	online	standby	ifStackStatus
10.224.0.110/11			ipAddressPrefixPrefi
fd73:738c:286:e000::6			ipAddressPrefixPrefi
fe80::200:a8ff:fe44:3			ipAddressPrefixPrefi
224.0.0.1			mgmdInverseHostCache
224.0.0.5			mgmdInverseHostCache
224.0.0.6			mgmdInverseHostCache
ff01::1			mgmdInverseHostCache
ff02::1			mgmdInverseHostCache
ff02::1:ff00:6e			mgmdInverseHostCache
ff02::1:ff44:3385			mgmdInverseHostCache
12	13	14	ifIndex
Link Multiplexor	82546 R3 Gigabi	82546 R3 Gigabi	ifDescr
9.20K	9.20K	9.20K	ifMtu
0000A8 443385	0000A8 443385	0000A8 643385	ifPhysAddress
up	up	up	ifOperStatus
0:03:03.067	0:02:21.068	0:02:24.054	ifLastChange
0:02:21.061	0:02:21.068	0:02:24.054	ifCounterDiscontinui
	fullDuplex	fullDuplex	dot3StatsDuplexStatu
0	1.00G	1.00G	ifHighSpeed
	602K	89.6K	ifInOctets
	1.19K	586	ifInUcastPkts
	161K	37.5K	ifOutOctets
	1.20K	587	ifOutUcastPkts
	4.27K		ifInMulticastPkts
	130	195	ifInBroadcastPkts
	633		ifOutMulticastPkts
	65		ifOutBroadcastPkts

%swsle#sdlmux.10.8.3	#enet.10.8.3	#enet.11.8.3	ifName
online	online	standby	ifStackStatus
10.0.244.110/24			ipAddressPrefixPrefi
fd73:738c:286:f4::6e/			ipAddressPrefixPrefi
fe80::200:a8ff:fe45:3			ipAddressPrefixPrefi
224.0.0.1			mgmdInverseHostCache
224.0.0.5			mgmdInverseHostCache
224.0.0.6			mgmdInverseHostCache
ff01::1			mgmdInverseHostCache
ff02::1			mgmdInverseHostCache
ff02::1:ff00:6e			mgmdInverseHostCache
ff02::1:ff45:3385			mgmdInverseHostCache
15	16	17	ifIndex
Link Multiplexor	82546 R3 Gigabi	82546 R3 Gigabi	ifDescr
9.20K	9.20K	9.20K	ifMtu

(Continued on next page)



0000A8 453385	0000A8 453385	0000A8 653385	ifPhysAddress
up	up	up	ifOperStatus
0:03:04.007	0:02:27.052	0:02:30.049	ifLastChange
0:02:27.046	0:02:27.052	0:02:30.049	ifCounterDiscontinui
	fullDuplex	fullDuplex	dot3StatsDuplexStatu
0	1.00G	1.00G	ifHighSpeed
	549K	50.0K	ifInOctets
	1.18K	584	ifInUcastPkts
	153K	37.4K	ifOutOctets
	1.17K	585	ifOutUcastPkts
	4.26K		ifInMulticastPkts
	12	77	ifInBroadcastPkts
	627		ifOutMulticastPkts
	65		ifOutBroadcastPkts

%swsle#sdlmux.10.7.0	#enet.10.7.0	#enet.11.7.0	ifName
online	online	standby	ifStackStatus
10.2.110.110/24			ipAddressPrefixPre
fd73:738c:286:26e::6e/64			ipAddressPrefixPre
fd73:738c:286:26e:200:a8ff:			ipAddressPrefixPre
fe80::200:a8ff:fe46:3385%8/			ipAddressPrefixPre
224.0.0.1			mgmdInverseHostCac
ff01::1			mgmdInverseHostCac
ff02::1			mgmdInverseHostCac
ff02::1:ff00:6e			mgmdInverseHostCac
ff02::1:ff46:3385			mgmdInverseHostCac
18	19	20	ifIndex
Link Multiplexor	82546 R3 Giga	82546 R3 Giga	ifDescr
9.20K	9.20K	9.20K	ifMtu
0000A8 463385	0000A8 463385	0000A8 663385	ifPhysAddress
up	up	up	ifOperStatus
0:03:04.046	0:02:35.043	0:02:40.026	ifLastChange
0:02:33.040	0:02:35.043	0:02:40.026	ifCounterDiscontin
	fullDuplex	fullDuplex	dot3StatsDuplexSta
0	1.00G	1.00G	ifHighSpeed
	37.5K	37.5K	ifInOctets
	584	584	ifInUcastPkts
	38.1K	37.4K	ifOutOctets
	585	585	ifOutUcastPkts
	2	3	ifInBroadcastPkts
	8		ifOutMulticastPkts
	1		ifOutBroadcastPkts

%swsle#sdlmux.10.7.1	#enet.10.7.1	#enet.11.7.1	ifName
online	online	standby	ifStackStatus
10.2.110.210/24			ipAddressPrefixPrefi
fe80::200:a8ff:fe47:3			ipAddressPrefixPrefi
224.0.0.1			mgmdInverseHostCache
ff01::1			mgmdInverseHostCache

(Continued on next page)

ff02::1			mgmdInverseHostCache
ff02::1:ff47:3385			mgmdInverseHostCache
21	22	23	ifIndex
Link Multiplexor	82546 R3 Gigabi	82546 R3 Gigabi	ifDescr
9.20K	9.20K	9.20K	ifMtu
0000A8 473385	0000A8 473385	0000A8 673385	ifPhysAddress
up	up	up	ifOperStatus
0:03:04.098	0:02:43.037	0:02:46.031	ifLastChange
0:02:43.028	0:02:43.037	0:02:46.031	ifCounterDiscontinui
	fullDuplex	fullDuplex	dot3StatsDuplexStatu
0	1.00G	1.00G	ifHighSpeed
	37.5K	37.5K	ifInOctets
	584	584	ifInUcastPkts
	37.7K	37.4K	ifOutOctets
	585	585	ifOutUcastPkts
	2	3	ifInBroadcastPkts
	3		ifOutMulticastPkts
	1		ifOutBroadcastPkts

%swsle#sdlmux.10.7.3	%swsle#enet.10.7.3	ifName
online	online	ifStackStatus
204.252.133.100/24		ipAddressPrefixPrefix
2001:1900:3008:fc85::64/64		ipAddressPrefixPrefix
fe80::200:a8ff:fe49:3385%10/6		ipAddressPrefixPrefix
224.0.0.1		mgmdInverseHostCacheAddres
ff01::1		mgmdInverseHostCacheAddres
ff02::1		mgmdInverseHostCacheAddres
ff02::1:ff00:64		mgmdInverseHostCacheAddres
ff02::1:ff49:3385		mgmdInverseHostCacheAddres
24	25	ifIndex
Link Multiplexor	82546 R3 Gigabit NI	ifDescr
9.20K	9.20K	ifMtu
0000A8 493385	0000A8 493385	ifPhysAddress
up	up	ifOperStatus
0:03:05.016	0:02:57.016	ifLastChange
0:02:57.009	0:02:57.016	ifCounterDiscontinuityTime
	fullDuplex	dot3StatsDuplexStatus
0	1.00G	ifHighSpeed
	1.77K	ifInOctets
	8	ifInUcastPkts
	4.18K	ifOutOctets
	37	ifOutUcastPkts
	12	ifInMulticastPkts
	10	ifOutMulticastPkts
	1	ifOutBroadcastPkts

%swsle#sdlmux.11.7.3	%swsle#enet.11.7.3	ifName
online	online	ifStackStatus
204.252.134.101/24		ipAddressPrefixPrefix

(Continued on next page)

2001:1900:3008:fc86::65/64				ipAddressPrefixPrefix
fe80::200:a8ff:fe4a:3385%11/6				ipAddressPrefixPrefix
224.0.0.1				mgmdInverseHostCacheAddress
ff01::1				mgmdInverseHostCacheAddress
ff02::1				mgmdInverseHostCacheAddress
ff02::1:ff00:65				mgmdInverseHostCacheAddress
ff02::1:ff4a:3385				mgmdInverseHostCacheAddress
26		27		ifIndex
Link Multiplexor		82546 R3 Gigabit NI		ifDescr
9.20K		9.20K		ifMtu
0000A8 4A3385		0000A8 4A3385		ifPhysAddress
up		up		ifOperStatus
0:03:05.050		0:03:00.013		ifLastChange
0:03:00.007		0:03:00.013		ifCounterDiscontinuityTime
		fullDuplex		dot3StatsDuplexStatus
0		1.00G		ifHighSpeed
		1.08K		ifInOctets
		5.11K		ifOutOctets
		48		ifOutUcastPkts
		12		ifInMulticastPkts
		9		ifOutMulticastPkts
		1		ifOutBroadcastPkts

## Related Information

See [Chapter 10](#) for related troubleshooting information.

## ngrep

The `ngrep` command, or network grep utility, is a network packet analyzer. The command is an open source application.

For information about this command, see online documentation (for example, <http://ngrep.sourceforge.net/>).

## omon

### Purpose

The `omon` command starts an Open Shortest Path First (OSPF) monitoring session that allows you to monitor OSPF on this and other routers. You use the `omon` command to obtain information from OSPF log and configuration files, add and delete static routes, and obtain general information about OSPF running on the router.

#### NOTE

OSPF uses the term router to refer to a system that has been configured to act as a router, that is, that can forward packets from one network to another. In this sense, when you enable IP forwarding, a module is a router. To use `omon`, you should make sure IP forwarding is enabled.

### Display Form

```
----- omon -----
-config_file: >system>stcp>omonconf
```

### Command-Line Form

```
omon [-config_file path_name]
```

### Arguments

- ▶ `-config_file path_name`  
Specifies the complete path name of the destination file. If you do not specify a path name, the command uses the `omonconf` file, in the `(master_disk)>system>stcp` directory, by default.

### Explanation

The `omon` command starts an OSPF monitoring session. In an OSPF monitoring session, you enter subcommands that specify the type of information to obtain.

Configuring a Destination File

Before you use the `omon` command, you create a destination file. In the destination file, you specify IP addresses of interfaces on the local OpenVOS module and on other modules or routers. You can monitor OSPF on the specified interfaces and on the router that contains the interface.

To create the destination file, you do the following:

- 1. Copy the template `omonconf` file (in the `(master_disk)>system>stcp>templates`) to the `(master_disk)>system>stcp` directory.
- 2. In the copy of the `omonconf` file in the `(master_disk)>system>stcp` directory, list the IP addresses and, optionally, the host names of interfaces on modules or routers that are part of the autonomous system (AS).

Place each interface on a separate line and use the following format for each entry:

*IP\_address host\_name*

Table 9-18 describes the *IP\_address host\_name* fields.

Table 9-18. Fields in a Destination File

Field	Description
<i>IP_address</i>	Specifies the Internet Protocol (IP) address of an interface on the router.
<i>host_name</i>	Specifies the host name of the interface on the router. Optional.

NOTES \_\_\_\_\_

- 1. To use the `omon` command to obtain information from interfaces on the local module and make changes to its `ospfd` daemon process, add entries to the destination file for the local interfaces.
- 2. You might be able to use the `omon` command to monitor some other implementations of OSPF. To determine whether you can do so, add an interface to the destination file and try the `omon` subcommands.

## Configuring an omon Session

Use the omon subcommands described in [Table 9-19](#) to configure your omon session.

**Table 9-19. The omon Subcommands: Configuring the Session**

Subcommand	Description
exit	Exits from the omon command.
list_destinations	Displays the list of destinations in the destination file. Use this information in the commands described in <a href="#">Table 9-20</a> .
list_history	Displays a numbered list of the commands you have entered. To repeat a command, you enter quotation marks (") followed by the number assigned to the command.
list_local_commands	Displays the commands in this table.
list_remote_commands	Displays the commands you can use to monitor OSPF on this and other routers. <a href="#">Table 9-20</a> describes these commands.
log_to_file file_name	Specifies that results from omon commands are logged to a file as well as standard output (typically, the terminal). The ospfd.io file, in the (master_disk) >system>stcp>temp directory, is a log of information returned to omon commands.  Use this subcommand to manage output on the local module.
log_to_stdio	Specifies that results from omon commands are only sent to standard output (typically, the terminal).

## Obtaining OSPF Information

In an omon session, you use the subcommands described in [Table 9-20](#) to obtain OSPF information about interfaces and modules or routers. Specify an interface in front of the subcommand, using one of the following formats:

- *IP\_address*

The IP address of an interface, in dotted decimal notation. The interface must be configured in the ospfdconf file on the local module.

- *@destination\_number*

The at sign (@) followed by a number that specifies an interface configured in the destination file. The number that identifies the first interface in the file is 1.

- *@*

The at sign (@) with no subsequent number specifies the previously specified destination.

**Table 9-20. The omon Subcommands: Monitoring Modules and Routers** (Page 1 of 4)

Subcommand	Description
<pre>add_static_route [IP_address] [subnet_mask] [forwarding_a ddress] [cost] [age] [type] [tag]  delete_static_route [IP_address] [subnet_mask] [forwarding_address] [cost] [age] [type] [tag]</pre>	<p>Adds or removes the specified static route as <i>external</i>, that is, imported from another AS.</p> <p>The <i>IP_address</i> argument specifies an interface.</p> <p>The <i>subnet_mask</i> argument specifies the subnet mask for the IP address of the interface.</p> <p>The <i>forwarding_address</i> argument specifies the next hop in the route.</p> <p>The <i>cost</i> argument specifies the metric for using the interface.</p> <p>The <i>age</i> argument specifies, in seconds, how long the static route is valid.</p> <p>The <i>type</i> argument specifies the type of the metric. A value of 1 indicates that to compute the cost for a path, the <i>ospfd</i> daemon should add internal AS link-state metrics to the metric value of <i>cost</i>. A value of 2 indicates that the <i>ospfd</i> daemon should use only the value of <i>cost</i> to compute the cost for a path.</p> <p>The <i>tag</i> argument specifies the AS from which you are exporting the route.</p>
<pre>display_MIB_type_data</pre>	<p>Displays the type of information that would be collected in an SNMP MIB (Simple Network Management Protocol Management Information Base).</p> <p><b>Note:</b> This version of STCP does not support the OSPF SNMP MIB; to obtain this information from a local interface or from an interface on another module running this version of STCP, use this <i>display_MIB_type_data</i> subcommand.</p>
<pre>enable_interface [IP_addr] no_enable_interface [IP_addr]</pre>	<p>Enables or disables the use of the interface specified by the <i>IP_addr</i> argument.</p>
<pre>event_logging no_event_logging</pre>	<p>Enables debug logging and logging of all events. <a href="#">Table 9-21</a> describes the events that are logged.</p> <p><b>Note:</b> The <i>pkt_logging</i> subcommand lets you enable logging of packets.</p>



**Table 9-20. The omon Subcommands: Monitoring Modules and Routers** (Page 2 of 4)

Subcommand	Description
flag [value]	Enables logging of the type of events specified by <i>value</i> . <a href="#">Table 9-21</a> describes the decimal values you can specify (you must specify decimal values). This subcommand also enables debug logging of the ospfd daemon process on the specified router.
host_route_information	Displays information from host entries in the ospfdconf file.
list_area_information	Displays the area ID and authentication type, whether the router imports AS-external type link-state advertisements (LSAs), the number of times the OSPF daemon has run the OSPF algorithm, the number of border routers and boundary routers in the area, the types of internal LSAs, and the default cost for interfaces added to the area.
list_database value	Displays a summary of the link-state database (LSD) on the specified router. The summary displays an entry for each LSA which includes an area ID, LSA type, link ID, the advertising router, age, length, sequence number, and cost. To also display the retransmission list, specify a value other than 0 (zero) for <i>value</i> . For more information about a specific LSA, use the list_lsa subcommand.
list_errors	Displays event information about OSPF packets and the number of times each type of event occurred.  <a href="#">Appendix D</a> describes the events listed by this command.
list_general_information	Displays information about the router that includes the router ID and its border router and boundary router status. OpenVOS modules always have a value of 0 for the boundary router status.
list_interfaces	Lists the interfaces used by the OSPF implementation on the specified router, and displays the designated router and backup designated router for each interface.
list_lsa [area_id] [type] [link_id] [router]	Displays information about the LSAs received by the router specified in front of the subcommand, from the router specified by the <i>router</i> argument.

**Table 9-20. The omon Subcommands: Monitoring Modules and Routers** (Page 3 of 4)

Subcommand	Description												
list_lsa (Cont.)	<p>The <i>area_id</i> argument specifies the area or areas for which LSA information is to be displayed. Specify BACKBONE, 0.0.0.0 (equivalent to BACKBONE), or an area ID in dotted notation. You can also specify a wildcard address that contains zeros (0) in the right-hand positions to match multiple areas. For example, to display LSAs associated with areas whose area number begins with 172.16, specify 172.16.0.0.</p> <p>The <i>type</i> argument specifies a number that identifies a type of LSAs to display; specify one of the following numbers:</p> <table> <tr> <th>Number</th><th>Link-State Advertisement Type</th></tr> <tr> <td>1</td><td>Router</td></tr> <tr> <td>2</td><td>Network</td></tr> <tr> <td>3</td><td>Summary IP advertisements</td></tr> <tr> <td>4</td><td>Boundary router summary link advertisements</td></tr> <tr> <td>5</td><td>AS external advertisements. For this type, do not specify an <i>area_id</i> argument.</td></tr> </table> <p>The <i>link_id</i> argument specifies the link ID, which identifies the source of the LSA, a 32-bit number in dotted decimal notation. If you do not specify a link ID, the command displays summary information about all LSAs, including link IDs. If you do specify a link ID, the command displays detailed information about the LSAs from the specified link ID. You can use wildcard addresses containing zeros (0) in the right-hand positions to specify multiple link IDs. For example, if you specify 172.16.0.0, the command displays all LSAs whose link ID begins with 172.16.</p> <p>The <i>router</i> argument specifies the IP address, in dotted decimal notation, of another router that supports OSPF.</p>	Number	Link-State Advertisement Type	1	Router	2	Network	3	Summary IP advertisements	4	Boundary router summary link advertisements	5	AS external advertisements. For this type, do not specify an <i>area_id</i> argument.
Number	Link-State Advertisement Type												
1	Router												
2	Network												
3	Summary IP advertisements												
4	Boundary router summary link advertisements												
5	AS external advertisements. For this type, do not specify an <i>area_id</i> argument.												
list_neighbors [value]	Lists the neighbors of the specified router. To also display the retransmission list, specify a value other than 0 for <i>value</i> .												
list_packet_types	<p>Displays a log of packet types sent and received by the specified router.</p> <p><b>Note:</b> The <i>pkt_logging</i> subcommand lets you enable logging of packets.</p>												
list_timer_queues	Displays the current timer queues on the specified router.												
list_ranges	Displays the address ranges configured in the <i>ospfdconf</i> file.												
list_routing_table	Displays the OSPF routing table on the specified router.												

**Table 9-20. The omon Subcommands: Monitoring Modules and Routers** (Page 4 of 4)

Subcommand	Description
list_tree	Displays the routing table tree on the specified router.
pkt_logging no_pkt_logging	Turns packet logging on or off. <a href="#">Table 9-21</a> lists the packet types that are logged. This subcommand also enables debug logging on the specified module. When you issue this subcommand, information about packets is logged to the ospfd.pkt file.  <b>Note:</b> The flag subcommand lets you enable logging of other events.
sort_database	Displays a summary of the LSD on the specified router, sorted by area and then by link ID.
terminate_ospfd	Stops the ospfd daemon process on the specified router.

## OSPFD Logging

Debug logs on the local OpenVOS module provide information you can use to troubleshoot the ospfd daemon process on the module. [Table 9-21](#) describes the values to use in the flag subcommand to specify the type of information you want to log and describes the information logged when you issue the pkt\_logging and event\_logging subcommands. (For information about debug logs, see [“Problem 6: Module Cannot Route Packets”](#) on page 10-6.)

**Table 9-21. Values for the flag Subcommand** (Page 1 of 2)

Decimal Value	Information Logged
1	Hello packets
2	Database description packets
4	Link-state request packets
8	Link-state update packets
16	Link-state acknowledged packets
31	All of the preceding packets
32	Building LSA event
64	Running the SPF algorithm event
128	Type of LSA being flooded event
256	Received LSA packets event
512	Received packets with bad type event

**Table 9-21. Values for the `flags` Subcommand** (Page 2 of 2)

Decimal Value	Information Logged
1024	Timer function training event
2048	State transitions event
4096	Packet dump event
65536	All input and output activity

## Examples

This section contains a sample destination file, an example of the `omon` command, and examples of some `omon` subcommands.

### Example 1: Sample Destination File

The following sample destination file lists three modules. On each line, the first value is the IP address of an interface on a module, and the second value, which is optional, is the host name of the module.

```
10.16.10.21      test.system.1
198.168.10.21   test.system.2
172.16.10.2     test.system.3
```

### Example 2: Starting an `omon` Session

The following command starts an `omon` command session and uses the default `omonconf` destination file:

```
omon
```

The command prompt that appears contains the number of the command and a description of the format for entering `omon` subcommands:

```
[ 0 ] dest command params >
```

You can use the number, in this case 0, to enter the command again.

### Example 3: Displaying Area Information

The following `omon` subcommand displays information about an area. The `@1` item specifies the first entry in the destination file.

```
@1 list_area_information
```

The command returns information similar to the following:

```
remote-command <b > sent to 10.16.10.21 1
      Source <<10.16.10.21      test.system.1>>
Area ID: 0.0.0.1
Auth Type: None  Import ASE: On  Spf Runs: 7
Local ABRs: 1  Local ASBRs: 1  Inter LSAs: 12  Inter Cksum sum: 0x87b07
```

#### Example 4: Listing Neighbors

The following `omon` subcommand lists the neighbors of the router specified by `@2`, the second entry in the destination file.

```
@2 list_neighbors
```

The command returns information similar to the following:

```
remote-command <N > sent to 192.168.10.21 1
      Source <<10.15.10.2 test.system.2>>
Area      Interface      Router Id      Nbr IP Addr      State      Mode      Pri
-----
0.0.0.1 192.168.10.21 192.168.10.1 192.168.10.17 Full Master 5
0.0.0.2 192.168.11.7 192.168.11.2 192.168.11.57 Full Master 5
```

#### Example 5: Listing Packet Types

The following `omon` subcommands start logging of packets and list the packets sent from and received by a router. The `@3` item specifies the third entry in the destination file.

```
@3 pkt_logging
@3 list_packet_typesFu
```

The command returns information similar to the following:

```
remote-command <c > sent to 172.16.10.2 1
      Source <<172.16.10.2 test.system.3>>
IO stats from: Wed Jul 19 10:07:06 2000
```

```
>> RECEIVED:
      20: Monitor request
      1426: Hello
      9: DB Description
      1: Link-State Req
      99: Link-State Update
      18: Link-State Ack
```

*(Continued on next page)*

```
>> SENT:
    2498: Hello
        5: DB Description
        4: Link-State Req
       21: Link-State Update
       91: Link-State Ack
```

### Example 6: Logging Hello Packets

The following command starts logging of all Hello packets from the first router in the destination file.

```
@1 flag 1
```

### Example 7: Listing LSAs

The following command displays information about router-LSAs received by the first router in the destination file for the router identified by 172.17.0.1 in area 0.0.0.0.

```
@1 list_lsa 0.0.0.0 1 172.17.0.1
```

The command returns information similar to the following:

```
remote-command <a 0.0.0.0 1 172.17.0.1 0.0.0.0 > sent to
172.17.10.21 1

      Source <<172.17.10.21      st21.100mb>>
LSA  type: RTR ls id: 172.17.0.1 adv rtr: 172.17.0.1 age: 1
     len: 36 seq #: 80001969 cksum: 0xec5
     Capabilities: As Border: On Area Border: On
     Link count: 1
     link id: 172.17.254.3 data: 172.17.254.1 type: Transit net
metric: 1
```

## Related Information

For more information, see the following documentation.

- [“The ospfdconf File and Routing Information” on page 5-31](#) describes the format of the ospfdconf file.
- [“Problem 6: Module Cannot Route Packets” on page 10-6](#) describes the debug log files.
- [“OSPF Routing” on page A-19](#) provides an overview of the Open Shortest Path First (OSPF) routing protocol.

# packet\_monitor

**Privileged**

## Purpose

The `packet_monitor` command displays information about packets that are transmitted or received through a network interface on the module. You must be a privileged user to issue this command.

## NOTES

1. To save information displayed by the `packet_monitor` command, use the `start_logging` and `stop_logging` commands. See the *OpenVOS Commands Reference Manual* (R098) for descriptions of the `start_logging` and `stop_logging` commands.
2. The `packet_monitor` command supports only IPv4 addresses. For IPv6 addresses, use `tcpdump`.
3. The `packet_monitor` command does not parse and display newer protocol features very well. For better packet formatting, use `tcpdump`.
4. The `packet_monitor` command cannot create or parse industry standard `pcap` files. To generate or parse `pcap` files, use `tcpdump`.

## Display Form

```
----- packet_monitor -----
-interface: 0
-count: 0
-hex_dump: no
-length: 128
-numeric: no
-time_stamp: no
-verbose: no
-pkt_hdr: no
-hex_header: no
-filter: yes
```

## Command-Line Form

```
packet_monitor [-interface string]
               [-count number]
               [-hex_dump]
               [-length data_bytes]
               [-numeric]
               [-time_stamp]
               [-verbose]
               [-pkt_hdr]
               [-hex_header]
               [-no_filter]
```

## Arguments

► *-interface string*

Specifies the network interface device through which the command listens to packet traffic. You can specify more than one interface. To do so, separate the names of the network interface devices with a space (*-interface interface1 interface2 interface3*). If you do not specify this argument, the command listens to packet traffic on all active network interface devices on your module. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation *-i* for the *-interface* argument.

### NOTICE

---

The `packet_monitor` command creates a queue for each attached interface. These queues use streams resources that are shared system wide. If a system has 10 interfaces and you start `packet_monitor` without using the `-interface` argument to specify an interface, the system uses 10 times the required amount of streams resources as it would if you had used the `-interface` argument to specify one network interface. The result can be the failure to create new connections or to transmit or receive data over existing connections. It is more efficient to monitor a single interface than multiple interfaces. (Some OpenVOS releases, however, can prevent too many invocations of `packet_monitor`.)



**NOTE** 

---

Do not issue the `packet_monitor` command interactively on an interface that is controlling a TELNET terminal. Doing so creates a positive feedback loop.

► `-count number`

Terminates command execution once the command receives the number of packets specified by *number*. If you do not specify this argument, the command uses the default value 0 and continues to execute until you cancel it. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-c` for the `-count` argument.

► `-hex_dump`

CYCLE

Displays the bytes of each packet (in hexadecimal notation) in addition to the default output displayed for each packet. You should specify the `-length` argument with this argument to limit the amount of data that is displayed. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-x` for the `-hex_dump` argument. By default, the command displays 128 bytes of each packet in addition to the default output displayed for each packet.

► `-length number`

Specifies the maximum number of bytes of data that are displayed. The default value is 128; the maximum value is 1642. You should specify this argument with the `-hex_dump` argument. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-l` for the `-length` argument.

► `-numeric`

CYCLE

Displays the host addresses and port numbers in standard dot-notation form rather than by symbolic name (the default representation). Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-n` for the `-numeric` argument.

**NOTICE** 

---

If you specify `-no_numeric` and your module is configured to query the DNS server, the `packet_monitor` command generates a high volume of network traffic. In the information displayed for each packet, the `packet_monitor` command must include the symbolic names of the source and destination hosts rather than the numeric IP addresses (since the command was issued with the `-no_numeric` argument). To determine the symbolic names, the `packet_monitor`

command queries the DNS server, which maps numeric IP addresses to symbolic names. The `packet_monitor` command must query the DNS server twice for each packet, once for the source address and once for the destination address.

- ▶ `-time_stamp` CYCLE  
Displays the time at which each packet was received. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-t` for the `-time_stamp` argument. By default, the command does not display a time stamp for each packet.
- ▶ `-verbose` CYCLE  
Displays additional fields of TCP and UDP packets. The source and destination host addresses (including port numbers) are displayed by default. Note that if you are issuing the command directly from command level without using the form, you can specify the abbreviation `-v` for the `-verbose` argument.
- ▶ `-pkt_hdr` CYCLE  
Displays the 14-byte Ethernet header for each Ethernet packet, which includes 6 bytes for the Destination Address, 6 bytes for the Source Address, and 2 bytes for the Protocol Type. By default, the command does not display the Ethernet header for each packet.
- ▶ `-hex_header` CYCLE  
Displays the 20-byte IP header in hexadecimal format. By default, the command does not display the IP header in hexadecimal format.
- ▶ `-no_filter` CYCLE  
Enables you to specify a filter for the packets. If you use the display form of the command, this argument is set to `yes` by default and automatically displays the `packet_monitor: specify filter` display form. If you change this argument to `no`, it suppresses the display of the `packet_monitor: specify filter` display form, but applies the conditions specified by default in the display form.

If you issue the command directly from command level instead of using the display form, specify the `-filter` argument with the appropriate filter arguments. You can specify the abbreviation `-f` for the `-filter` argument.

If you use the display form of the command with the default setting of `yes` for the `-filter` argument, the `packet_monitor: specify filter` display form appears, as follows:

```
----- packet_monitor: specify filter -----
-host:          █
-dst:
-src:
-protocol:
-receive:       yes          -transmit:       yes
-arp:           no           -ip:             no
-mac_host:      -mac_dst:
-mac_src:      -mac_broadcast: no
-ip_broadcast: no          -port:           0
-src_port:      0          -dst_port:       0
-filter:        no
```

Note that the `packet_monitor: specify filter` display form also includes the `-filter` argument. Each time you specify a filter with the `-filter` argument, another `packet_monitor: specify filter` display form appears, which enables you to specify an additional filter.

See [Arguments of packet\\_monitor: specify filter](#) (below) for information about the arguments of the `packet_monitor: specify filter` display form. See [Using Filters](#) for additional information about using the `-no_filter` argument and specifying filters.

## Arguments of packet\_monitor: specify filter

By default, the `packet_monitor: specify filter` display form limits the search to packets received and transmitted over the network interface on the module. Specify any additional filtering with the following arguments.

### ► -host *string*

Determines whether the value of the `destination` or `source` field of the IP header in the packet matches the value you specify. The value you specify must have the standard dot-notation form (for example, `172.16.10.10`) or be a symbolic name (for example, `m1.mktg.corp.com`).

If you specify the `-host` filter argument, you cannot specify the `-dst` or `-src` filter argument.

### ► -dst *string*

Determines whether the value of the `destination` field of the IP header in the packet matches the value you specify for the destination. The value you specify

must have the standard dot-notation form (for example, 172.16.10.10) or be a symbolic name (for example, aberdaron.test.corp.com).

If you specify the `-dst` filter argument, you cannot specify the `-src` or `-host` filter argument.

► `-src string`

Determines whether the value of the `source` field of the IP header in the packet matches the value you specify for the source. The value you specify must have the standard dot-notation form (for example, 172.16.10.10) or be a symbolic name (for example, aberdaron.test.corp.com).

If you specify the `-src` filter argument, you cannot specify the `-dst` or `-host` filter argument.

► `-protocol string`

CYCLE

Determines whether the packet is an IP packet of the type you specify. Specify one of the following values: `ip`, `icmp`, `tcp`, or `udp`.

► `-no_receive`

CYCLE

Limits the search to packets received (from the LAN) through a network interface device on your module.

► `-no_transmit`

CYCLE

Limits the search to packets transmitted (to the LAN) through a network interface device on your module.

► `-arp`

CYCLE

Detects and displays only ARP protocol packets. By default, the command detects and displays all types of packets, including ARP protocol packets.

► `-ip`

CYCLE

Detects and displays only IP protocol packets. By default, the command detects and displays all types of packets, including IP protocol packets.

► `-mac_host string`

Determines whether the value of the `source` or `destination` field of the IP header in the packet matches the value you specify. The value you specify must be a MAC address (for example, 00:00:a8:00:80:cd).

► `-mac_dst string`

Determines whether the value of the `destination` field of the IP header in the packet matches the value you specify. The value you specify must be a MAC address (for example, 00:00:a8:02:3b:7f).

If you specify the `-mac_dst` filter argument, you cannot specify the `-mac_src` or `-mac_host` filter argument.

**NOTE**

In general, you should not specify both a MAC address (with the `-mac_dst` argument) and an IP address (with the `-dst` argument) in the same `packet_monitor`:  
`specify filter display form`.

▶ `-mac_src string`

Determines whether the value of the `source` field of the IP header in the packet matches the value you specify. The value you specify must be a MAC address (for example, `00:00:a8:00:28:a9`).

If you specify the `-mac_src` filter argument, you cannot specify the `-mac_dst` or `-mac_host` filter argument.

▶ `-mac_broadcast`**CYCLE**

Detects and displays MAC broadcast packets only. By default, the command detects and displays all types of packets, including MAC broadcast packets.

▶ `-ip_broadcast`**CYCLE**

Detects and displays IP broadcast packets only. By default, the command detects and displays all types of packets, including IP broadcast packets.

▶ `-port number`

Determines whether the packet's source-port or destination-port value matches the port number you specify. The port number you specify can be the name or number of a well-known TCP port, as defined in the database file `(master_disk)>system>stcp>services`. The default value is 0, which matches all ports; 0 filters no ports.

If you specify the `-port` filter argument, you cannot specify the `-src_port` or `-dst_port` filter argument.

▶ `-src_port number`

Determines whether the packet's source-port value matches the port number you specify. The port number you specify can be the name or number of a well-known TCP port, as defined in the file `(master_disk)>system>stcp>services`. The default value is 0, which matches all ports; 0 filters no ports.

If you specify the `-src_port` filter argument, you cannot specify the `-dst_port` or `-port` filter argument.

▶ `-dst_port number`

Determines whether the packet's destination-port value matches the port number you specify. The port number can be the name or number of a well-known TCP port, as defined in the database file

(master\_disk) >system>stcp>services. The default value is 0, which matches all ports; 0 filters no ports.

If you specify the `-dst_port` filter argument, you cannot specify the `-src_port` or `-port` filter argument.

► `-filter`

CYCLE

Invokes another `packet_monitor: specify filter` display form. By default, this argument is set to `no`, which suppresses the display of an additional `packet_monitor: specify filter` display form.

## Explanation

By default, the `packet_monitor` command listens to and displays information about the packet traffic passing through the active network interface devices on your module. Specify the `-interface` argument to display traffic information for a specific network interface.

The `packet_monitor` command accepts UNIX-style abbreviations for many of the arguments shown in the main display form.

When the `packet_monitor` command detects a fragmented Internet packet, it marks each fragment with an asterisk (\*), starting with the second packet that the command encounters in each Internet packet (the marked fragment would look like *\*fragment\**).

If you issue the `packet_monitor` command without the `-verbose` argument, the command only displays the source and destination host addresses (including the port numbers).

## Using Filters

You can define several filters in a single `packet_monitor` command by specifying the `-filter` argument more than once. The command attempts to match a packet with each filter in the sequence in which the filters were specified. If a packet does not match any of the filters, the packet is ignored.

When a packet matches a filter, the `packet_monitor` command duplicates the packet and stores it on the input queue, from which the packet can be displayed. When a packet matches a filter and is stored in memory, it is not compared to the other filters specified in the `packet_monitor` command.

You can issue more than one `packet_monitor` command and specify a different filter in each command. In this case, each packet is compared with the filter specified in each command. Therefore, a packet that matches more than one filter is stored for subsequent display on the input queue of each `packet_monitor` command whose filter selected the packet.

Generally, it is more efficient to issue one `packet_monitor` command and specify the `-filter` argument with several filters (for example, `-host` and `-protocol`), rather than issue several `packet_monitor` commands and specify one filter in each command.

If you specify conflicting filter arguments in a filter, the `packet_monitor` command is unable to match packets. For example, if you specify both the `-arp` and `-ip` filter arguments in a filter, the command does not match either packet type.

The `-receive` and `-transmit` filter arguments define the scope of the `packet_monitor` command most broadly. The other filter arguments narrow the scope of packet matching. For example, the `-receive` argument detects all packets received from remote hosts on the network. Subsequently, the `-ip` argument narrows the match to IP packets only. The `-host` argument further narrows the match to IP packets received from the specified remote host.

When specifying filter arguments, consider the conflicts that can result from duplicating or overlapping arguments. For example, specifying the `-dst` filter argument achieves the same result as specifying both the `-transmit` and `-host` arguments, as long as the `destination` and `remote` values are identical.

Although it may be easier to specify several filters by using successive `packet_monitor: specify filter` display forms, you can specify several filters using the command-line form. Using the command-line form, you delimit each successive filter with the `-filter` argument. That is, a filter specification begins with the `-filter` argument and ends with next occurrence of the `-filter` argument. The examples that follow show how you specify filters using the command-line form.

If you specify multiple `-filter` arguments, the command searches for packets that match the first filter you specify. If it finds a packet that matches the first filter, it simply display that information and does not attempt to match the second filter. The command only tries to match the second filter if it does not find a packet that matches the first filter.

## Examples

This section shows several examples of the `packet_monitor` command. The output of each example includes the following fields.

- The `dir` field indicates the direction (for example, `T` for transmit and `R` for receive).
- The `len` field indicates the data length, not including the packet header.
- The `proto` field indicates the protocol.
- The `source` field indicates the host that is the source of the packets.
- The `destination` field indicates the destination host.
- The `icmp type` field identifies whether ICMP packets are echo requests or echo replies.

- The `src port` field identifies the source port.
- The `dst port` field identifies the destination port.
- The `tcp type` field indicates the flags that are set in the TCP protocol header.

### Example 1

In the following example, the user is logged in on the local host `m1-1.ad.corp.com`. The command searches for packets sent to or received from the host `psych.ad.corp.com`. The `-count` argument value is 10, which enables the command to search for 10 packets.

The command output shows that TCP packets were sent to the host `psych.ad.corp.com` from `m1-1.ad.corp.com` and that the TCP packets were received by `m1-1.ad.corp.com` from the host `psych.ad.corp.com`. The packets match the filter argument `-host psych.ad.corp.com`.

```
packet_monitor -interface #e1.1 -count 10 -filter -host psych.ad.corp.com
```

dir	len	proto	source	destination	icmp type	src port	dst port	tcp type
T 0027		TCP	m1-1.ad.corp.com	psych.ad.corp.com	telnet		1951	PA
R 0000		TCP	psych.ad.corp.com	m1-1.ad.corp.com		1951		telnet A
T 0066		TCP	m1-1.ad.corp.com	psych.ad.corp.com	telnet		1951	PA
R 0000		TCP	psych.ad.corp.com	m1-1.ad.corp.com		1951		telnet A
T 0150		TCP	m1-1.ad.corp.com	psych.ad.corp.com	telnet		1951	PA
R 0000		TCP	psych.ad.corp.com	m1-1.ad.corp.com		1951		telnet A
T 0150		TCP	m1-1.ad.corp.com	psych.ad.corp.com	telnet		1951	PA
R 0000		TCP	psych.ad.corp.com	m1-1.ad.corp.com		1951		telnet A
T 0076		TCP	m1-1.ad.corp.com	psych.ad.corp.com	telnet		1951	PA
R 0000		TCP	psych.ad.corp.com	m1-1.ad.corp.com		1951		telnet A

### Example 2

The following example reduces the `-count` value to 5 and shows the output that results from the Example 1 `packet_monitor` command when you specify the `-verbose` argument to display additional information. (The plus sign (+) is a line-continuation character, and is not entered as part of the command.)

```
packet_monitor -interface #e1.1 -count 5 -verbose -filter
+-host psych.ad.corp.com
```

dir	len	proto	source	destination	icmp type	src port	dst port	tcp type
Rcvd IP	Ver/HL 45, ToS 0, Len 29, ID 1399, Flg/Frg 0, TTL 3c, Prtl 6							
	Cksum f9ec, Src 866f3240, Dst 866f322b							
TCP from	psych.ad.corp.com.1956 to m1-1.ad.corp.com.telnet							
	seq 24562157, ack 26102183, window 4096, 1. data bytes, flags Push Ack.							
	X/Off 05, Flags 18, Cksum 40d6, Urg-> 0000							

*(Continued on next page)*



```

Xmit IP   Ver/HL 45, ToS 0, Len 28, ID 3ed1, Flg/Frg 0, TTL 1e, Prtl 6
          Cksum ec5, Src 866f322b, Dst 866f3240
TCP from m1-1.ad.corp.com.telnet to psych.ad.corp.com.1956
  seq 26102183, ack 24562157, window 33170, 0. data bytes, flags Ack.
  X/Off 05, Flags 10, Cksum 2bdf, Urg-> 0000

Xmit IP   Ver/HL 45, ToS 0, Len 47, ID 3ed4, Flg/Frg 0, TTL 1e, Prtl 6
          Cksum ec93, Src 866f322b, Dst 866f3240
TCP from m1-1.ad.corp.com.telnet to psych.ad.corp.com.1951
  seq 1923553, ack 141154189, window 16384, 31. data bytes, flags Push Ack.
  X/Off 05, Flags 18, Cksum ee00, Urg-> 0000

Rcvd IP   Ver/HL 45, ToS 0, Len 29, ID 139e, Flg/Frg 0, TTL 3c, Prtl 6
          Cksum f9e7, Src 866f3240, Dst 866f322b
TCP from psych.ad.corp.com.1956 to m1-1.ad.corp.com.telnet
  seq 24562157, ack 26102183, window 4096, 1. data bytes, flags Push Ack.
  X/Off 05, Flags 18, Cksum ebd4, Urg-> 0000

Rcvd IP   Ver/HL 45, ToS 0, Len 28, ID 13a2, Flg/Frg 0, TTL 3c, Prtl 6
          Cksum f9e4, Src 866f3240, Dst 866f322b
TCP from psych.ad.corp.com.1951 to m1-1.ad.corp.com.telnet
  seq 141154189, ack 1923553, window 4096, 0. data bytes, flags Ack.
  X/Off 05, Flags 10, Cksum 285d, Urg-> 0000

```

### Example 3

The following example shows a filter that limits the search to ICMP protocol packets.

```
packet_monitor -interface #e1.1 -count 3 -filter -host m13 -protocol icmp
```

dir	len	proto	source	destination	icmp type	src port	dst port	tcp type
T	64	ICMP	m5.corp.com	m13.corp.com	echo rep			
R	64	ICMP	m13.corp.com	m5.corp.com	echo			
T	64	ICMP	m5.corp.com	m13.corp.com	echo rep			

### Example 4

The following example shows a packet\_monitor command that uses the -pkt\_hdr argument to display the contents of an Ethernet header.

```
packet_monitor -interface #e1.1 -pkt_hdr
```

dir	len	proto	source	destination	icmp type	src port	dst port	tcp type
T	Ether	Dst	00:00:a8:8a:80:4b	Src 00:00:a8:8a:07:05	Type 0800	(IP)		
0004	TCP	m5-2.corp.com		m1.corp.com		16300	1101	S
T	Ether	Dst	00:00:a8:8a:80:4b	Src 00:00:a8:8a:07:05	Type 0800	(IP)		
0004	TCP	m5-2.corp.com		m1.corp.com		16301	1101	S

*(Continued on next page)*

```
R Ether Dst 00:00:a8:8a:07:05 Src 00:00:a8:8a:80:4b Type 0800 (IP)
0003 TCP m1.corp.com m5-2.corp.com 1101 16300 PA

R Ether Dst 00:00:a8:8a:07:05 Src 00:00:a8:8a:80:4b Type 0800 (IP)
0003 TCP m1.corp.com m5-2.corp.com 1101 16300 PA
R Ether Dst 00:00:a8:8a:07:05 Src 00:00:a8:8a:80:4b Type 0800 (IP)BREAK
```

Example 5

The following example shows a `packet_monitor` command that uses the `-hex_header` argument to display the contents of the IP header in hexadecimal format. (The `+` character in the output indicates a line continuation.)

```
packet_monitor -interface #e1.1 -hex_header -count 2

dir                                icmp type      tcp
len proto source                destination    src port dst port type
R   offset 0 . . . 4 . . . 8 . . . C . . . 0...4... 8...C...
    0 45 0 0 28 b8 40 0 0 3c 6 ef a0 86 6f 65 b E ( _`088@ _`08<
+<_`08o_`08 >oe<
    10 86 6f 65 5 >oe<
0000 TCP m1.corp.com m5-2.corp.com 1101 16393 FA

T   offset 0 . . . 4 . . . 8 . . . C . . . 0...4... 8...C....
    0 45 0 0 28 75 cc 0 0 3c 6 32 15 86 6f 65 5 E (u_`08L _`08<
+<2<>oe<
    10 86 6f 65 b >oe<
0000 TCP m5-2.corp.com m1.corp.com 16393 1101 A
```

Related Information

See [Chapter 10](#) for related troubleshooting information. See also the [netstat](#) command description, which appears earlier in this chapter.

# ping

## Purpose

The `ping` command enables you to determine the network status of a remote host or hosts by sending ICMP (for IPv4) or ICMP6 (for IPv6) Echo Request packets to the hosts and listening for Echo Reply packets. The command displays the number of packets sent and received. By default, the command transmits four Echo packets containing 64 bytes of data (a periodic uppercase sequence of alphabetic characters).

The command offers the arguments shown in the following display form, unless the `(master_disk)>system>stcp` directory contains an empty file called `.flash_ping`. (If the `.flash_ping` file exists, the command offers additional arguments, as shown later in this command description.)

## Display Form

```
----- ping -----
host: 
-node_query: 
-pattern: 
-source_addr: 
-multicast_interface: 
-ipv6:          yes          -ipv4:          yes
-count:         4            -packetize:     64
-timeout:       15
-preload: 
-fragment:      yes
-full_unicast_mtu: no
-probe_node_info_mcast: no
-verbose:       no
-beep:          no
-run_forever:   no
-debug:         no
-socket_buffers: 
-numeric:       no
-full_multicast_mtu: no
-node_info:     no
-once:          no
-beep_on_drops: no
-silent_return: no
-node_dns:      no
```

## Command-Line Form

```
ping host
    [-no_ipv6]
    [-no_ipv4]
    [-node_query addr_type]
    [-count number]
    [-packetsize number]
    [-timeout number]
    [-hoplimit number]
    [-preload number]
    [-socket_buffers integer]
    [-pattern string]
    [-source_addr value]
    [-no_fragment]
    [-numeric]
    [-multicast_interface string]
    [-full_unicast_mtu]
    [-full_multicast_mtu]
    [-probe_node_info_mcast]
    [-node_info]
    [-verbose]
    [-once]
    [-beep]
    [-beep_on_drops]
    [-run_forever]
    [-silent_return]
    [-debug]
    [-node_dns]
```

## Arguments

- ▶ *host* **Required**

The remote host whose availability you want to verify. Specify an IPv4 or IPv6 address of the host or its symbolic host name, as specified in the `hosts` database file or found by a DNS server. If you do not specify a host, the command prompts you to specify one. You can also specify several hosts to ping, separating each IP address or host name with a space.
- ▶ *-node\_query addr\_type*

Generates the query *ICMPv6 Node Information Node Addresses*, rather than an echo-request. An ICMPv6 Node Information Node Addresses query asks the peer

about the addresses the other end supports. For *addr\_type*, specify one or more of the following characters:

- a Requests unicast addresses from all of the responder's interfaces. If *a* is omitted, only those addresses belonging to the interface that has the responder's address are requested.
- c Requests responder's IPv4-compatible and IPv4-mapped addresses.
- g Requests responder's global-scope addresses.
- s Requests responder's site-local addresses.
- l Requests responder's link-local addresses.
- A Requests responder's anycast addresses. Without *A*, the responder returns unicast addresses, only. With *A*, the responder returns anycast addresses, only. Note that the specification does not specify how to get responder's anycast addresses.

- ▶ `-pattern string`  
Enables `ping` to fill in the data portion of packets with pad bytes. For *string*, specify a pattern in 1 to 16 bytes. If *host* is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-source_addr value`  
Specifies the source address of the request packets, where *value* is a host parameter that must match one of the IP addresses assigned to an interface using the `ifconfig` command.
- ▶ `-no_ipv6` CYCLE  
Specifies that the *host* argument cannot be an IPv6 address. By default (the value *yes*), the *host* argument can be an IPv6 address.
- ▶ `-no_ipv4` CYCLE  
Specifies that the *host* argument cannot be an IPv4 address. By default (the value *yes*), the *host* argument can be an IPv4 address.
- ▶ `-count number`  
Enables you to specify the number of packets to be transmitted. The default is 4. The maximum number of packets you can send is 32,767.
- ▶ `-packetsize number`  
Enables you to specify the size of the packets transmitted. The default is 64 bytes.
- ▶ `-timeout time`  
Specifies the amount of time, in seconds, that the command will wait for a response before continuing. The default is 15 seconds.

- ▶ `-hoplimit number`  
Specifies the number of network segments over which a packet can travel before a router discards it. For IPv6, this setting is the hop limit. For IPv4, this setting is the time to live (TTL).
- ▶ `-preload number`  
Specifies the number of packets the `ping` command sends as quickly as possible before reverting to normal operation.
- ▶ `-socket_buffers integer`  
Sets the sizes of the send and receive socket buffers, using the `SO_SNDBUF` and `SO_RCVBUF` socket-level options.
- ▶ `-fragment` CYCLE  
Disables fragmentation of packets. By default (the value `yes`), `ping` allows packet fragmentation.
- ▶ `-numeric` CYCLE  
Disables the command from converting IP addresses to domain names, so that the output displays only IP addresses. By default (the value `no`), the output includes domain names converted from IP addresses.
- ▶ `-multicast_interface string`  
Enables `ping` to send packets to multicast addresses. For IPv4, *string* is one of the IP addresses assigned to the interface from which multicast packets are sent. For IPv6, *string* is the SDLMUX device name for the interface from which multicast packets are sent.
- ▶ `-full_unicast_mtu` CYCLE  
Enables `ping` to send unicast packets that fit into the full IPv6 MTU. By default (the value `no`), unicast packets fit into the minimum IPv6 MTU. If *host* is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-full_multicast_mtu` CYCLE  
Enables `ping` to send multicast packets that fit into the full IPv6 MTU. By default (the value `no`), multicast packets fit into the minimum IPv6 MTU. If *host* is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-probe_node_info_mcast` CYCLE  
Enables `ping` to send probe packets to the node information multicast group (`ff02::2:xxxx:xxxx`).
- ▶ `-node_info` CYCLE  
Generates the query *ICMPv6 Node Information supported query types*, which asks the peer what kind of queries it supports. If *host* is an IPv4 address, `ping` ignores this argument and displays a warning message.

- ▶ `-verbose` CYCLE  
Enables output to display packets other than ICMP (for IPv4) or ICMP6 (for IPv6) Echo Responses. By default, the command displays only ICMP (for IPv4) or ICMP6 (for IPv6) Echo Responses.
- ▶ `-once` CYCLE  
Enables `ping` to stop sending packets after receiving a single reply packet. If `host` is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-beep` CYCLE  
Enables `ping` to beep with each received packet. If `host` is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-beep_on_drops` CYCLE  
Enables `ping` to beep with each dropped packet. If `host` is an IPv4 address, `ping` ignores this argument and displays a warning message.
- ▶ `-run_forever` CYCLE  
Sends the ping packets to the destination host indefinitely (forever). By default, the command does not send the packets to the destination host indefinitely.

#### NOTICE

If you specify the `-run_forever` argument, the `ping` command will send a ping packet as soon as the last ping packet is answered or times out. As a result, the network will be flooded with ping packets.

- ▶ `-silent_return` CYCLE  
Suppresses the display of information about return packets. By default, the command displays information about return packets.
- ▶ `-debug` CYCLE  
Sets the `SO_DEBUG` socket-level option, which enables debugging in the underlying protocol modules.
- ▶ `-node_dns` CYCLE  
Generate the query *ICMPv6 Node Information DNS Name*. If `host` is an IPv4 address, `ping` ignores this argument and displays a warning message.

## Using Additional Arguments by Creating a `.flash_ping` File

A system administrator can create an empty file called `.flash_ping` in the `(master_disk)>system>stcp` directory to enable the `ping` command to offer additional arguments: `-record_route`, `-loose_route`, and `-strict_route`. If no `.flash_ping` file exists, these arguments do not appear in the display form of the

command and are not available. If the file exists, the display form of the command appears as follows:

```

host:
- node_query:
- pattern:
- source_addr:
- ipv6: yes - ipv4: yes
- count: 4 - packet_size: 64
- timeout: 15 - hop_limit:
- preload: - socket_buffers:
- fragment: yes - numeric: no
- full_unicast_mtu: no - full_multicast_mtu: no
- probe_node_info_mcast: no - node_info: no
- verbose: no - once: no
- beep: no - beep_on_drops: no
- run_forever: no - silent_return: no
- debug: no - node_dns: no
- record_route: - loose_route: no
- strict_route: no

```

- -record route *number*

If the file `(master_disk)>system>stcp>.flash_ping` exists, this argument affects the IP header of the outgoing ICMP message by storing the route of the outgoing packet, and that followed by the returning ICMP Echo Reply packet, in the `RECORD_ROUTE` field. You specify a maximum number of routes that are recorded using the `number` argument. If you specify the `-record_route` argument, you cannot specify the `-loose_route` or `-strict_route` arguments. You can record up to nine routes through routers/gateways. By default, the command does not record routes.

- -loose route

CYCLE

If the file `(master_disk)>system>stcp>.flash_ping` exists, this argument affects the IP header of the outgoing ICMP message by routing the packet via the list of hosts. Consecutive hosts may be separated by intermediate routers/gateways. In other words, the packet is *loose source routed*, which means that the sender specifies a list of IP addresses that the packet must traverse, but the packet can also pass through other routers between any two addresses in the list. You can specify up to nine intermediate hosts. If you specify the `-loose_route` argument, you cannot specify the `-record_route` or `-strict_route` arguments. By default, the command does not enable the packet to be loose-source routed.



► `-strict_route`

CYCLE

If the file `(master_disk)>system>stcp>.flash_ping` exists, this argument affects the IP header of the outgoing ICMP message by routing the packet via the list of hosts, but consecutive hosts may not be separated by intermediate routers/gateways. In other words, the packet is *strict source routed*, which means that the sender specifies the exact path that the IP packet must follow. If a router encounters a next hop in the source route that is not on a directly connected network, an ICMP `source route failed` error message is returned. You can specify up to nine intermediate hosts. If you specify the `-strict_route` argument, you cannot specify the `-record_route` or `-loose_route` arguments. By default, the command does not enable the packet to be strict source routed.

## Explanation

In its simplest form, the `ping` command enables you to verify that you are able to reach a specific host. It confirms basic IP connectivity to the specified host.

If the command fails completely, it may return the following message.

```
ping: No reply. Time Out!
```

## Examples

This section provides several examples of the `ping` command.

### Example 1

The following example of the `ping` command verifies that the host with IP address 172.16.2.30 is operational by sending three ICMP Echo Request packets.

```
ping 172.16.2.30 -count 3
```

```
Pinging host 172.16.2.30 : 172.16.2.30
ICMP Echo Reply:TTL 254 time = 6 ms
ICMP Echo Reply:TTL 254 time = 6 ms
ICMP Echo Reply:TTL 254 time = 4 ms
Host 172.16.2.30 replied to all 3 of the 3 pings
```

**Example 2**

The following example of the `ping` command verifies that the host with IPv6 address `fd73:738c:286:1503::85` is operational by sending four ICMP Echo Request packets.

```
ping fd73:738c:286:1503::85
Pinging host fd73:738c:286:1503::85
ICMP6 Echo Reply: icmp_seq=0 hlim=4 time=0.307 ms
ICMP6 Echo Reply: icmp_seq=1 hlim=4 time=0.798 ms
ICMP6 Echo Reply: icmp_seq=2 hlim=4 time=0.133 ms
ICMP6 Echo Reply: icmp_seq=3 hlim=4 time=0.099 ms
Host fd73:738c:286:1503::85 replied to all 4 of the 4 pings
```

**Example 3**

The following example of the `ping` command verifies that hosts `victor` and `hugo` are operational.

```
ping victor hugo

Pinging host victor.sw.comp.com (victor) : 172.16.2.40
ICMP Echo Reply:TTL 254 time = 16 ms
ICMP Echo Reply:TTL 254 time = 15 ms
ICMP Echo Reply:TTL 254 time = 17 ms
ICMP Echo Reply:TTL 254 time = 14 ms
Host victor.sw.comp.com replied to all 4 of the 4 pings
Pinging host hugo.sw.comp.com (hugo) : 172.16.2.25
ICMP Echo Reply:TTL 254 time = 5 ms
ICMP Echo Reply:TTL 254 time = 6 ms
ICMP Echo Reply:TTL 254 time = 5 ms
ICMP Echo Reply:TTL 254 time = 5 ms
Host hugo.sw.comp.com replied to all 4 of the 4 pings
```

### Example 4

The following example of the `ping` command indicates that one of the specified hosts could be reached (m5), but one could not be reached (m10).

```
ping m5 m10 -count 5

Pinging host m5.corp.com (m5) : 172.16.12.2
ICMP Echo Reply:TTL 60 time = 6 ms
ICMP Echo Reply:TTL 60 time = 6 ms
ICMP Echo Reply:TTL 60 time = 6 ms
ICMP Echo Reply:TTL 60 time = 6 ms
ICMP Echo Reply:TTL 60 time = 6 ms
Host m5.corp.com replied to all 5 of the 5 pings
Pinging host m10.corp.com (m10) : 172.16.13.2
ping getmsg: Network is unreachable.
```

### Example 5

The following example uses the `-record_route` argument (available when an empty `.flash_ping` file exists) to record nine routes through routers/gateways. Note that each route is separated in the output by a colon (:).

```
ping -record_route 9 192.232.7.100

Pinging host 192.232.7.100 : 192.232.7.100
ICMP Echo Reply:TTL 237:DF set time = 363 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.1.114:4.0.1
+.245)
ICMP Echo Reply:TTL 237:DF set time = 661 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.2.34:4.0.1.
+245)
ICMP Echo Reply:TTL 237:DF set time = 360 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.2.34:4.0.1.
+245)
ICMP Echo Reply:TTL 237:DF set time = 533 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.2.34:4.0.1.
+245)
Host 192.232.7.100 replied to all 4 of the 4 pings
```

**Example 6**

The following example of the `ping` command verifies that IPv6 multicast packets sent from interface `%sq3#sdlmux.qa974.10-6.1.11-6.1` get responses. It does this by sending four ICMP Echo Request packets to the specified multi-cast address (`ff02::1` is the IPv6 multi-cast address for all hosts).

```
ping -multicast_interface %sq3#sdlmux.qa974.10-6.1.11-6.1
ff02::1
Pinging host ff02::1
ICMP6 Echo reply came from fe80::200:a8ff:fe40:3a98%2:
+icmp_seq=0 hlim=64 time = 4.669 ms (DUP!)
ICMP6 Echo reply came from fe80::200:a8ff:fe40:3a98%2:
+icmp_seq=1 hlim=64 time=12.193 ms (DUP!)
ICMP6 Echo reply came from fe80::200:a8ff:fe40:8110%2:
+icmp_seq=0 hlim=64 time=12.507 ms (DUP!)
ICMP6 Echo reply came from fe80::200:a8ff:fe40:8110%2:
+icmp_seq=1 hlim=64 time=12.297 ms (DUP!)
Host ff02::1 replied to 0 of the 4 pings
Other hosts generated 4 or more replies to 4 pings
ready 11:15:00
```

**Example 6**

The following example shows a `ping` command that times out.

```
ping -record_route 9 192.232.7.100

Pinging host 192.232.7.100 : 192.232.7.100
ICMP Echo Reply:TTL 237:DF set time = 455 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.2.34:4.0.1.
+245)
ICMP Echo Reply:TTL 237:DF set time = 886 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.1.114:4.0.1.
+.245)
ICMP Echo Reply:TTL 237:DF set time = 462 ms
Received RECORD ROUTE:(172.16.18.5:stratus-inet-gw.stratus.com:stratus-inet-d
+mz.stratus.com:199.94.204.42:206.34.78.39:4.0.1.122:4.0.1.178:4.0.2.34:4.0.1.
+245)
ping: No reply. Time Out !!
```

## Command Status

The `ping` command returns a `command_status` value.

For packets sent to a unicast address:

- If at least one response is received from that host, the value 0 is returned.
- If no responses are received from that host, the value 1 is returned.

For packets sent to a multicast address:

- If any responses are received from any host, the value 0 is returned.
- If no responses are received, the value 1 is returned.

If an error is reported, `command_status` will be either 1 or the error code.

## Related Information

See [“The `hosts` File and Host Name Information” on page 5-7](#) for more information about entries for hosts. In [Chapter 6](#), see [“Modifying and Executing the `start\_stcp.cm` Command Macro” on page 6-4](#) and [“Verifying STCP Configuration” on page 6-12](#) for information about starting and verifying STCP.

## route

***Privileged***

### Purpose

The `route` command enables you to manually control the routes in the network routing table. Use the `route` command to display routes or to add, delete, modify, or monitor a route.

### Display Form

```
----- route -----
command:          █
destination:
gateway:
subnet_mask:
-ipv6:            yes
-ipv4:            yes
-mtu:
-expire:
-weight:
-ask:             yes
-flush:           no
-reject:          no
-blackhole:      no
-default_gateway: no
-numeric:         no
-long:            no
-verbose:         no
```

## Command-Line Form

```
route [ command ]
      [ destination ]
      [ gateway ]
      [ subnet_mask ]
      [ -no_ipv6 ]
      [ -no_ipv4 ]
      [ -mtu number ]
      [ -expire seconds ]
      [ -weight number ]
      [ -no_ask ]
      [ -flush ]
      [ -reject ]
      [ -blackhole ]
      [ -default_gateway ]
      [ -numeric ]
      [ -long ]
      [ -verbose ]
```

## Arguments

### ► *command*

CYCLE

The command (route operation) that you want to issue. You can specify one of these route operations:

- **print**—Looks up a destination in the routing table if *destination* is specified; otherwise, it displays all entries in the routing table.
- **add**—Adds a route.
- **delete**—Deletes a route.
- **change**—Modifies a route.
- **monitor**—Continuously reports changes in the routing table.

By default, the command does not perform a route operation. If you do not specify a route operation, you must specify the `-default_gateway` argument.

### ► *destination*

The symbolic name, IPv4 address, or IPv6 address of the destination network or host. If you specify a symbolic name, the information must be defined in the database file `(master_disk)>system>stcp>networks` or `(master_disk)>system>stcp>hosts`. The destination is the target of the specified route.

You can use CIDR notation by specifying */N* after the name, where *N* is referred to as the prefix length and indicates the number of high-order 1 bits in the netmask. The default prefix length for IPv6 is 64. The default prefix length for IPv4 remains unfixed. See the [Explanation](#) in the `ifconfig` command description for more information about prefixes.

If you include */N* when specifying a name for the *destination* argument, you cannot specify the *subnet\_mask* argument.

If you do not specify a prefix length with *destination* or you do not specify *subnet\_mask*, the `route` command uses a host route mask. For IPv4, the host route mask is 255.255.255.255 and the host route prefix length is /32 (CIDR or slash notation). For more information about configuring subnets and a subnet mask, see [“Specifying a Network Mask to Identify Subnets” on page A-13](#).

- ▶ *gateway*  
The symbolic name, IPv4 address, or IPv6 address of the network router/gateway to which packets are addressed in order to reach *destination*. If you specify a symbolic name, you must also define the routers/gateways in the database file (master\_disk)>system>stcp>hosts. If you specify an IP address, both *gateway* and *destination* must be IPv6 addresses or both must be IPv4 addresses.
- ▶ *subnet\_mask*  
Enables you to specify a mask for the configuration of subnets. (A *subnet* is a logical subnetwork). This argument is valid only when *destination* is an IPv4 address. This argument is not valid when *destination* includes CIDR notation.
- ▶ `-no_ipv6` CYCLE  
Specifies that the *destination* argument cannot be an IPv6 address. By default (the value *yes*), the *destination* argument can be an IPv6 address.
- ▶ `-no_ipv4` CYCLE  
Specifies that the *destination* argument cannot be an IPv4 address. By default (the value *yes*), the *destination* argument can be an IPv4 address.
- ▶ `-mtu number`  
Specifies an unsigned integer to override the hardware's MTU or packet size for the route to *destination*.
- ▶ `-expire seconds`  
Specifies the time in seconds at which the route to *destination* expires. The minimum value is 1.
- ▶ `-weight number`  
Specifies the weight of the route. The value *number* corresponds to the parameter `ipRouteMetric1` in the SNMP route MIB. The default value is 1 if the route is



created using a routing socket or `-1` if the route is created using SNMP. The routing protocol used determines the meaning of weight.

- ▶ `-no_ask` CYCLE  
 Suppresses the display of a prompt asking you to verify that you want to delete (flush) all previously established routers/gateways from the routing table, as specified by the `-flush` argument. By default, the command prompts you to verify that you want to delete all previously established routers/gateways from the routing table.
- ▶ `-flush` CYCLE  
 Deletes (flushes) all previously established routers/gateways from the routing table when you add a route (that is, when *command* is `add`). The `route` command deletes the routing table entries, based on the specified `-ipv4` and `-ipv6` arguments, before adding a route. By default, the command does not delete routing table entries.
- ▶ `-reject` CYCLE  
 Prevents (rejects) packets from being routed. A limited number of ICMP or ICMPv6 packets then reply to inform the sender that the packet was not forwarded.
- ▶ `-blackhole` CYCLE  
 Prevents packets from being routed and does not inform the sender that the packets were dropped.
- ▶ `-default_gateway` CYCLE  
 Enables the configuration of a default router/gateway when *command* is `add`. All packets for which no destination exists in the router/gateway table are passed to the default router/gateway, when it exists. You can specify one default router/gateway. If you add a router/gateway as a default router/gateway, do not specify an IP address for *destination*; instead, specify an asterisk (\*).

You cannot specify this argument if *destination* is an IPv6 address and *command* is `add`, `change`, or `delete`.

IPv6 automatically finds routers; however, you may need to specify a default gateway interface for IPv6 when a configuration includes multiple routers. The `-default_ip6_interface` argument of the `ifconfig` command controls selection of the default IPv6 gateway interface. You can also control the selection using configuration parameters in the routers.

- ▶ `-numeric` CYCLE  
 Enables the `route` command to display only the IP address of the route to *destination*, which improves performance when *command* is `print` or `monitor`. By default (the value `no`), the command displays both IP addresses and host names.

- ▶ `-long` CYCLE  
Enables the `route` command to display detailed information on multiple lines when `command` is `print`. The detailed information includes `route to`, `destination`, `gateway`, `interface`, and more. By default (the value `no`), the command displays routing information (Network Address and Gateway Address/Interface) on single lines.
- ▶ `-verbose` CYCLE  
Enables the `route` command to display traffic on the routing socket used to communicate routing information to and from the kernel.

## Explanation

You can use the `route` command to display or monitor the routing table or to add, delete, or modify a route. You can also clear the routing table. STCP supports network routes (with subnet masks), host routes, and default routes through a default router/gateway.

To STCP, a *route* is a path by which data flows between two hosts. Simple networks may have only a single route between hosts. Complex networks may have multiple routes between hosts. In the latter case, some routes may offer lower response times, higher capacity, or lower costs. Some hosts may only be accessible via certain gateways.

When modules are not on the same subnets, you must explicitly define routes using the `route` command. Only one route can be defined to a given destination.

However, STCP automatically creates a route for the local network/subnetwork, based on the network address and subnet mask of a logical interface, when you configure the interface using the STCP `ifconfig` command. If, for example, you have two modules (modules `m1` and `m2`), where each module has a physical interface (which is associated with a logical interface) connected to network A and a different physical interface (which is associated with a logical interface) connected to network B, STCP automatically creates a route between the `m1` and `m2` logical interfaces to network A, and STCP automatically creates a different route between the `m1` and `m2` logical interfaces to network B.

To define routes so that they are recognized automatically when STCP is started, specify the appropriate number of `route` commands in the `start_stcp.cm` command macro. The template `start_stcp.cm` command macro provides a sample `route` command commented out. (See [“Modifying and Executing the start\\_stcp.cm Command Macro” on page 6-4](#) for a description of the `start_stcp.cm` command macro.)

Routes that you add using the `route` and `ifconfig` commands are called *static routes*. You cannot add a static route that duplicates another static route.

When sending packets, STCP looks for the most-specific route (that is, a route to a particular host) first and the least-specific route (that is, the default route, which requires only the specification of a local gateway to handle routing to remote networks and subnets) last. Routes to hosts are beneficial when you want to handle communication to a particular remote host differently from communication to other remote hosts.

The `print` operation enables you to print the routing table, which contains an entry for a default router/gateway, if one has been configured. This operation is essentially the same as issuing the `netstat` command with the `-routing` argument. If an ICMP Redirect request has been received for a given destination, the routing table indicates the router/gateway that will forward packets to that destination in the `Redirect` field. The `Life` field shows how long route will remain in effect. The time starts at 5 minutes and counts down to 0, when the route is deleted.

Note that if you do not specify both a destination and router/gateway when adding or changing a route, the command displays a message reminding you to do so.

To set the IP multicast *default system interface* (that is, the interface to send multicasts over if no interface has been specified via the `IP_MULTICAST_IF` option), issue the `route` command. This sets 224.0.0.0 to the interface IP address. For example:

```
route add 224.0.0.0 134.111.56.157
```

## NOTES

1. If you specify the IP multicast default system interface, you must use a local IP unicast address as the router/gateway. You cannot specify an alias as an IP multicast interface.
2. If you do not specify a local IP unicast interface, the application should use `INADDR_ANY`, which is equivalent to specifying 0.0.0.0 as the local IP unicast interface.

After you set the default system interface, you cannot change it unless you stop all IP multicast applications (that is, you must close all connections).

To avoid interpretation errors, always issue the `route` command with a completely specified IP address in *destination*. In the following example, the command correctly interprets the route to be 1.2.3.0.

```
route add 1.2.3.0 255.255.255.0
```

However, if you issue the command with an incomplete address in *destination*, the command interprets the route incorrectly. In the following example, the command interprets the route as 1.2.0.3.

```
route add 1.2.3 255.255.255.0
```

## Examples

The `route` command examples in this section are based on the following seven configured interfaces, as displayed in output of the `ifconfig` command:

```
%swsle#loop.m122: <UP, LOOPBACK, RUNNING, MTU:16384>
ether: 000000 000000
127.0.0.1 netmask 0xff000000
::1/128
fe80::1%1/64
nd6 options: add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.6.0: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 40bd4b
10.10.1.1 netmask 0xffffffff00 broadcast 10.10.1.255
fe80::200:a8ff:fe40:bd4b%2/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
%swsle#sdlmux.10.2.2: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 45bd4b
172.16.12.1 netmask 0xffffffff00 broadcast 172.16.12.255
fd73:738c:286:f4:200:a8ff:fe45:bd4b/64 auto_configure
fe80::200:a8ff:fe45:bd4b%3/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.2.0: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 42bd4b
134.111.75.177 netmask 0xffffffff00 broadcast 134.111.75.255
fe80::200:a8ff:fe42:bd4b%4/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.2.1: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 43bd4b
10.0.1.122 netmask 0xffffffff00 broadcast 10.0.1.255
fe80::200:a8ff:fe43:bd4b%5/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

(Continued on next page)

```
%swsle#sdlmux.11.2.1: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 44bd4b
10.0.2.122 netmask 0xffffffff00 broadcast 10.0.2.255
fe80::200:a8ff:fe44:bd4b%6/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal

%swsle#sdlmux.10.2.3: <UP, BROADCAST, RUNNING, NOFORWARDBROADCAST,
+KEEPALIVE, MTU:1500>
ether: 0000a8 46bd4b

172.16.3.28 netmask 0xffffffff00 broadcast 172.16.3.255
fd73:738c:286:e000:200:a8ff:fe46:bd4b/64 auto_configure
fe80::200:a8ff:fe46:bd4b%7/64
nd6 options: use_router_adv add_def_rtr neighbor_detect auto_linklocal
```

This section presents the following examples of the `route` command:

- [“Example 1: Defining a Default Router/Gateway and Verifying the Addition” on page 9-147](#)
- [“Example 2: Adding Routes and Verifying the Additions” on page 9-148](#)
- [“Example 3: Deleting a Route and Verifying the Deletion” on page 9-149](#)
- [“Example 4: Changing a Route and Verifying the Change” on page 9-150](#)
- [“Example 5: Defining Routes and Masks for a VLSN Configuration” on page 9-150](#)
- [“Example 6: Displaying Expiration Times” on page 9-151](#)
- [“Example 7: Using the `-numeric` Argument” on page 9-152](#)
- [“Example 8: Setting the Default System Interface” on page 9-153](#)

### Example 1: Defining a Default Router/Gateway and Verifying the Addition

In the following example, a `route` command adds router/gateway 172.16.12.30 as the default router/gateway for any destination not specified (indicated by the \*). A second `route` command displays the default router/gateway in the routing table.

```
route add * 172.16.12.30 -default_gateway

route print

Network Address                                     Gateway Address/Interface
default                                              172.16.12.30
10.0.1.0/24                                         %swsle#sdlmux.10.2.1
10.0.1.122                                         %swsle#loop.m122
10.0.2.0/24                                         %swsle#sdlmux.11.2.1
10.0.2.122                                         %swsle#loop.m122
10.10.1.0/24                                       %swsle#sdlmux.10.6.0
```

*(Continued on next page)*

```

10.10.1.1                                %swsle#loop.m122
127.0.0.0/8                             %swsle#loop.m122
134.111.75.0/24                         %swsle#sdlmux.10.2.0
swslem122.eng.stratus.com                %swsle#loop.m122
172.16.12.0/24                          %swsle#sdlmux.10.2.2
172.16.12.1                             %swsle#loop.m122
::                                       fe80::a6ba:dbff:fe56:f2ef%3
::1                                     %swsle#loop.m122
fd73:738c:286:f4::/64                  %swsle#sdlmux.10.2.2
fd73:738c:286:f4:200:a8ff:fe45:bd4b     %swsle#loop.m122
fe80::%1/64                            %swsle#loop.m122
fe80::%2/64                            %swsle#sdlmux.10.6.0
fe80::%3/64                            %swsle#sdlmux.10.2.2
fe80::%4/64                            %swsle#sdlmux.10.2.0
fe80::%5/64                            %swsle#sdlmux.10.2.1
fe80::%6/64                            %swsle#sdlmux.11.2.1
fe80::1%1                              %swsle#loop.m122
ff01::%1/32                            %swsle#loop.m122
ff01::%2/32                            %swsle#sdlmux.10.6.0
ff01::%3/32                            %swsle#sdlmux.10.2.2
ff01::%4/32                            %swsle#sdlmux.10.2.0
ff01::%5/32                            %swsle#sdlmux.10.2.1
ff01::%6/32                            %swsle#sdlmux.11.2.1
ff02::%1/32                            %swsle#loop.m122
ff02::%2/32                            %swsle#sdlmux.10.6.0
ff02::%3/32                            %swsle#sdlmux.10.2.2
ff02::%4/32                            %swsle#sdlmux.10.2.0
ff02::%5/32                            %swsle#sdlmux.10.2.1
ff02::%6/32                            %swsle#sdlmux.11.2.1

```

## Example 2: Adding Routes and Verifying the Additions

In the following example, the local host (172.16.12.1) is configured with three `route` commands to add the following:

- 172.16.12.10 as a router/gateway for destinations on network 172.16.13.0
- 172.16.12.20 as a router/gateway for destinations on network 172.16.14.0
- 172.16.12.30 as the router/gateway for the host 172.16.15.5

The first two commands configure network routes and specify the appropriate subnet masks; the third command configures a host route and does not specify a subnet mask. A fourth `route` command displays the routing table with the new entries. This assumes that the local host has multiple (at least three) routers/gateways available on its local subnet.

```

route add 172.16.13.0 172.16.12.10 255.255.255.0
route add 172.16.14.0 172.16.12.20 255.255.255.0
route add 172.16.15.5 172.16.12.30
route print

```

*(Continued on next page)*

```

Network Address                                Gateway Address/Interface
default                                         172.16.12.30
.
. [ lines removed ]
.
swslem122.eng.stratus.com                      %swsle#loop.m122
172.16.12.0/24                                %swsle#sdlmux.10.2.2
172.16.12.1                                  %swsle#loop.m122
172.16.13.0/24                                172.16.12.10
172.16.14.0/24                                172.16.12.20
172.16.15.5                                  172.16.12.30
::                                              fe80::a6ba:dbff:fe56:f2ef%3
::1                                             %swsle#loop.m122
fd73:738c:286:f4::/64                         %swsle#sdlmux.10.2.2
fd73:738c:286:f4:200:a8ff:fe45:bd4b          %swsle#loop.m122
.
. [ lines removed ]
.

```

### Example 3: Deleting a Route and Verifying the Deletion

The following example deletes the 172.16.15.5 route and verifies the deletion with another `route` command.

```

route delete 172.16.15.5
route print

```

```

Network Address                                Gateway Address/Interface
default                                         172.16.12.30
.
. [ lines removed ]
.
swslem122.eng.stratus.com                      %swsle#loop.m122
172.16.12.0/24                                %swsle#sdlmux.10.2.2
172.16.12.1                                  %swsle#loop.m122
172.16.13.0/24                                172.16.12.10
172.16.14.0/24                                172.16.12.20
::                                              fe80::a6ba:dbff:fe56:f2ef%3
::1                                             %swsle#loop.m122
fd73:738c:286:f4::/64                         %swsle#sdlmux.10.2.2
fd73:738c:286:f4:200:a8ff:fe45:bd4b          %swsle#loop.m122
.
. [ lines removed ]
.

```

**Example 4: Changing a Route and Verifying the Change**

The following example changes the 172.16.14.0 route to use router/gateway 172.16.12.40, a fourth router/gateway, instead of 172.16.12.20 (as shown in the previous example), and verifies the change with another `route` command.

```
route change 172.16.14.0 172.16.12.40 255.255.255.0
route print
```

Network Address	Gateway Address/Interface
default	172.16.12.30
.	
. [ lines removed ]	
.	
swslem122.eng.stratus.com	%swsle#loop.m122
172.16.12.0/24	%swsle#sdlmux.10.2.2
172.16.12.1	%swsle#loop.m122
172.16.13.0/24	172.16.12.10
172.16.14.0/24	172.16.12.40
::	fe80::a6ba:dbff:fe56:f2ef%3
::1	%swsle#loop.m122
fd73:738c:286:f4::/64	%swsle#sdlmux.10.2.2
fd73:738c:286:f4:200:a8ff:fe45:bd4b	%swsle#loop.m122
.	
. [ lines removed ]	
.	

**Example 5: Defining Routes and Masks for a VLSN Configuration**

In the following example, local Stratus host 172.16.3.28 needs network routes through host 172.16.3.17 (acting as a router) to reach the 172.16.4.0 subnet and the 172.16.4.128 subnet. Since the 172.16.4.0 subnet uses the subnet mask 255.255.255.128 and the 172.16.4.128 subnet uses the subnet mask 255.255.255.192, the `route` commands must be issued as follows:

```
route add 172.16.4.0 172.16.3.17 255.255.255.128
route add 172.16.4.128 172.16.3.17 255.255.255.192
route print
```

Network Address	Gateway Address/Interface
default	172.16.12.30
.	
. [ lines removed ]	
.	

*(Continued on next page)*



```

swslem122.eng.stratus.com          %swsle#loop.m122
172.16.3.0/24                      %swsle#sdlmux.10.2.3
172.16.3.28                        %swsle#loop.m122
172.16.4.0/25                      172.16.3.17
172.16.4.128/26                   172.16.3.17
172.16.12.0/24                    %swsle#sdlmux.10.2.2
172.16.12.1                       %swsle#loop.m122
172.16.13.0/24                    172.16.12.10
172.16.14.0/24                    172.16.12.40
::                                 fe80::a6ba:dbff:fe56:f2ef%3
::1                                %swsle#loop.m122
fd73:738c:286:f4::/64             %swsle#sdlmux.10.2.2
fd73:738c:286:f4:200:a8ff:fe45:bd4b %swsle#loop.m122
.
. [ lines removed ]
.

```

### Example 6: Displaying Expiration Times

In order to display expiration times, specify the `-long` argument, as in the following example.

```

route print -long

route to: default
destination: default
gateway: 204.79.142.1
interface: %tel4#sdlmux.10.6.0
flags: <UP,GATEWAY,DONE,STATIC>
weight: 1
mtu: 1500
.
. [ lines removed ]
.
route to: 127.0.0.0
destination: 127.0.0.0
mask: 255.0.0.0
interface: %tel4#loop.m4
flags: <UP,DONE,PINNED>
weight: 1
mtu: 16384

```

*(Continued on next page)*

```
route to: quantum.stratus.com
destination: quantum.stratus.com
gateway: 204.79.142.2
interface: %tel4#sdlmux.10.6.0
        flags: <UP,GATEWAY,HOST,DYNAMIC,DONE,PROTO3>
        weight: 1
        mtu: 1500
        expire: 14-08-06 16:51:36 edt

route to: nitro.stratus.com
destination: nitro.stratus.com
gateway: 204.79.142.2
interface: %tel4#sdlmux.10.6.0
        flags: <UP,GATEWAY,HOST,DYNAMIC,DONE,PROTO3>
        weight: 1
        mtu: 1500
        expire: 14-08-06 16:53:00 edt

.
. [ lines removed ]
.
```

### **Example 7: Using the -numeric Argument**

The following example shows output when the -numeric argument is specified.

```
route print -long -numeric

route to: default
destination: default
gateway: 204.79.142.1
interface: %tel4#sdlmux.10.6.0
        flags: <UP,GATEWAY,DONE,STATIC>
        weight: 1
        mtu: 1500

.
. [ lines removed ]
.
route to: 127.0.0.0
destination: 127.0.0.0
        mask: 255.0.0.0
interface: %tel4#loop.m4
        flags: <UP,DONE,PINNED>
        weight: 1
        mtu: 16384
```

*(Continued on next page)*

```

route to: 134.111.1.5
destination: 134.111.1.5
gateway: 204.79.142.2
interface: %tel4#sdlmux.10.6.0
flags: <UP,GATEWAY,HOST,DYNAMIC,DONE,PROTO3>
weight: 1
mtu: 1500
expire: 14-08-06 16:51:36 edt

route to: 134.111.18.9
destination: 134.111.18.9
gateway: 204.79.142.2
interface: %tel4#sdlmux.10.6.0
flags: <UP,GATEWAY,HOST,DYNAMIC,DONE,PROTO3>
weight: 1
mtu: 1500
expire: 14-08-06 16:53:02 edt
.
. [ lines removed ]
.
```

### Example 8: Setting the Default System Interface

The following example shows how to set the *default system interface* (that is, the interface over which multicasts are sent if no interface was specified in the `IP_MULTICAST_IF` option).

```
route add 224.0.0.0 172.16.12.15
```

## Related Information

See the routing information in [“Specifying Routes” on page 6-9](#) and [“STCP Routes” on page A-18](#).

For more information about the `IP_MULTICAST_IF` option, see the description of the `setsockopt` function in the *OpenVOS STREAMS TCP/IP Programmer’s Guide* (R420).

## rtsol

### Purpose

The `rtsol` command calls the `rtssold` daemon process.

Though this command is not privileged, it requires modify access to the RAW (IP) and UDP protocol devices.

### Display Form

```
----- rtsol -----
interface:
-autoprobe:      no
-configure:      no
-mobile_node:    no
-add_ra_src_address: no
-pid_file:
-other_script:
-resolvconf_script:
-debug:          off
```

### Command-Line Form

```
rtsol [ interface(s)
      [-autoprobe]
      [-configure]
      [-mobile_node]
      [-add_ra_src_address]
      [-pid_file PID_file_name]
      [-other_script path_name]
      [-resolvconf_script path_name]
```

### Arguments

The `rtsol` command is a command macro that calls the `rtssold` daemon process, including (automatically) the `rtssold -foreground` and `-single_probe` arguments. See the description of the `rtssold` daemon process for information about the arguments, since the arguments are identical.

## set\_udp\_ordering

### Purpose

The `set_udp_ordering` command specifies the type of ordering of UDP datagrams for all network components

### Display Form

```
----- set_udp_ordering -----  
udp_ordering:
```

### Command-Line Form

```
set_udp_ordering [udp_ordering]
```

### Arguments

► *udp\_ordering*

**CYCLE**

**Required**

Specifies the type of ordering, as follows:

- `loose`—UDP datagrams may be reordered for better performance.
- `strict`—STCP does as much as possible to keep UDP datagrams with the same ports and IP addresses from being reordered with the module. The `strict` policy forces a single CPU to handle all UDP traffic, resulting in a severe performance penalty.

### Explanation

The `set_udp_ordering` command specifies the type of ordering of UDP datagrams that contain identical source IP addresses, destination IP addresses, source ports, or destination ports for all network components (the Ethernet driver, SDLMUX, ARP, IP, IPsec, UDP, and so on). The ordering does not apply to UDP packets with different addresses or ports.

The `set_udp_ordering` command affects two areas of packet processing:

- After an Ethernet adapter is initialized, the command affects which hardware input and output queues packets are put on. You can change the ordering policy only if you take an interface out of service and bring it back into service.
- The command affects how messages are queued for IPsec processing. The command affects IPsec processing immediately.

The `set_udp_ordering` command must exist in one of the following files:

- `module_start_up.cm`—If the command exists in the `module_start_up.cm` file, the `set_udp_ordering` command line must appear before the `module_start_up.cm` file invokes the `start_stcp.cm` command macro.
- `start_stcp.cm`—If the command exists in the `start_stcp.cm` file, the `set_udp_ordering` command line must appear before the `start_stcp.cm` file invokes any `dlnux_admin` commands.

## Examples

The following command specifies that UDP datagrams be reordered for better performance:

```
set_udp_ordering loose
```

The following command specifies that STCP keep UDP datagrams with the same ports and IP addresses from being reordered with the module, to the extent possible:

```
set_udp_ordering strict
```

## Related Information

The following sections provide additional information about STCP and UDP:

- [“Defining the UDP Driver” on page 4-4](#)
- [“The protocols File and Protocol Information” on page 5-40](#)

## **setkey**

The `setkey` command adds, updates, dumps, or flushes Security Association Database (SAD) and Security Policy Database (SPD) entries in the kernel. For information about this command, see online BSD documentation (for example, <http://www.freebsd.org>).

## tcpdump

### Purpose

The `tcpdump` command provides a command-line packet analyzer.

The `tcpdump` command links to the `libpcap` library, which exists as the shared library `libpcap.so.1.4.0` in the `(master_disk)>system>lib` directory.

For a complete description of the `tcpdump` command, see <http://www.tcpdump.org/>. Note, however, the following exceptions in the OpenVOS implementation:

- The loopback device cannot be monitored.
- There is no kernel support for packet filtering.
- Promiscuous modes do not put the hardware in promiscuous mode; instead, the packet analyzer receives all packets that are received by the hardware the way it is configured by the other software.
- OpenVOS does not support monitor mode.
- OpenVOS does not use hardware checksum acceleration.
- The `-i (--interface)` option does not accept the value `any`.



# telnet

## Purpose

The `telnet` command begins a TELNET client session. You use the TELNET protocol to log in to a remote host.

## Display Form

```
----- telnet -----  
host: █  
port_number:  
-address: ipv4  
-lookup: yes
```

## Command-Line Form

```
telnet [host]  
      [port_number]  
      [-address type]  
      [-no_lookup]
```

## Arguments

► *host*

**Required**

Specifies the host name or IP address of the remote host with which you want to open a connection. For an IP address, specify the address type set by the `-address` argument (`ipv4` or `ipv6`).

If you specify a host, TELNET tries to open a connection. Once the connection is established, you are in TELNET input mode. If you do not specify a host, TELNET begins the session without opening a connection and places you in TELNET subcommand mode.

► *port\_number*

Specifies the TELNET command port on the remote host. Specify a port number only if you specify a value for the *host* argument. The default port number is 21.

**NOTICE**

The standard STCP TELNET server process (`telnetd`) and the MST server process (`telnet_msd`) must not be listening for requests at the same network port. If you execute both the `telnet` command and the `telnet_msd` command, you must specify a different value for the port argument of each command (`-network_port` for the `telnet_msd` command and `port_number` for the `telnet` command).

**► -address *type*****CYCLE**

Specifies the type of IP address, in standard dot notation. Values are:

- `ipv4` (the default)—The format of the IP address is IPv4, which consists of a 32-bit number in four octets.
- `ipv6`—The format of the IP address is IPv6, which consists of a 128-bit number in eight groups of four hexadecimal digits, separated by colons.

**► -lookup****CYCLE**

The default value `yes` enables the TELNET client to interpret the value of the `host` argument as a name (rather than as an IP address). Then, the TELNET client can use name services to look up the name to find its associated IP address. With the value `no`, the TELNET client does not look up the value of the `host` argument.

## Explanation

The `telnet` command invokes the TELNET client and starts a TELNET session. You can start a TELNET session by trying to connect to a remote host, or you can enter TELNET subcommand mode and then issue TELNET subcommands. Once you are connected to a remote host, you can log in and perform operations (you are in input mode at this point).

If you issue the `telnet` command and specify a host, a connection request is sent to the remote host. You can optionally specify a port number to access a specific TELNET service other than the default TELNET service. (The default TELNET service is a login service at port 21.) For example, to access a TELNET service at IP address `172.16.3.46`, you would specify the following:

```
telnet 172.16.3.46 2105
```

In the preceding example, the host associated with IP address `172.16.3.46` must advertise the service in its `services` database file at port number `2105`. If the connection request is accepted, the escape character (by default, `[Ctrl-]`) is displayed, and you are then prompted to log in to the host.

If you do not specify a host, you will see the `telnet>` prompt, which indicates that you are in TELNET subcommand mode.

In TELNET subcommand mode, you can open a connection by using the `open` subcommand or issue other TELNET subcommands.

#### NOTE

If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in TELNET services.

[Table 9-22](#) lists the TELNET subcommands in alphabetical order.

**Table 9-22. TELNET Subcommands**

Subcommand	Description
<code>? or help [subcommand]</code>	Provides information about the supported TELNET subcommands.
<code>close</code>	Closes a connection with the remote host.
<code>display [setting]</code>	Displays the current settings for input requests and operating modes.
<code>open host [port_number]</code>	Tries to open a connection with a remote host.
<code>quit</code>	Ends the TELNET session.
<code>send input_request</code>	Issues one or more input requests.
<code>set input_request key_sequence</code>	Defines key sequences for issuing input requests directly from input mode.
<code>status</code>	Displays status information for the current session.
<code>toggle operating_mode</code>	Turns the operating mode on or off. To see a list of the operating modes, issue the <code>toggle</code> subcommand with the <code>?</code> argument.

## Examples

The following example shows the beginning of a TELNET session.

```
telnet open 172.16.3.46
```

```
Trying...
```

```
Connected to 172.16.3.46.
```

```
Escape character is '^]'.
```

## Error Messages

The following table lists some common error messages associated with the `telnet` command.

Error Message	Description
<code>telnet: Unable to connect to remote host: Connection refused.</code>	The TELNET server on the remote host is not running.
<code>telnet: Unable to connect to remote host: Timeout period has expired.</code>	The remote host did not respond to a connection request before the timeout period elapsed. The remote host may not be in service.
<code>host: Unknown host.</code>	The specified host is not known to the system. You may want to try using the host's IP address instead.

## Related Information

See “[The `hosts.allow` and `hosts.deny` Files and TCP Wrappers](#)” on page 5-10 for information about TCP wrappers functionality.

See the *OpenVOS STREAMS TCP/IP User's Guide* (R421) for more information about using TELNET.

## telnet\_admin

**Privileged**

### Purpose

The `telnet_admin` command enables you to add, delete, modify, or list a local TELNET service while the `telnetd` daemon process is running on the module. Use this command to easily add or delete local login services or any incoming slave services, or to change the options associated with a service. (The `telnet-service` database file initially defines only the default login service as privileged on port 23.) This command automatically updates the `telnet-service` and `services` database files, and the changes stay in effect until you subsequently alter the service entry. It frees you from having to update the `telnet-service` and `services` database files each time you add, delete, or modify a local TELNET service. Note, however, that when you enter the `telnet_admin` command for an existing service, you **must** include an argument and value that corresponds to each field in the service's entry in the `telnet-service` file (see the [Explanation](#) for more information).

### Display Form

```
----- telnet_admin -----
command:      add
-service:
-device_prefix:
-keepalive:   yes
-linger:
-nodelay:     yes
-description:
-login:        yes
-privileged:  yes
-service_file: %sys#module>system>stcp>telnet-service
-services_port:
-local_ip:
```

## Command-Line Form

```
telnet_admin command
    [-service string]
    [-device_prefix string]
    [-no_keepalive]
    [-linger [number]]
    [-no_nodelay]
    [-description string]
    [-no_login]
    [-no_privileged]
    [-service_file path_name]
    [-services_port number]
    [-local_ip IP_address]
```

## Arguments

► *command*

**CYCLE**

**Required**

The TELNET operation that you want to perform. Specify one of the following operations.

- Specify **add** (the default on the display form) to add a TELNET login or incoming slave service to both the `telnet-service` and `services` database files.
- Specify **delete** to delete a TELNET service from the `telnet-service` and `services` database files.
- Specify **modify** to change an existing service entry.
- Specify **list** to view the entries for existing TELNET services in the `telnet-service` file, along with their corresponding port numbers.

If you issue the `telnet_admin` command from command level, you must specify an operation. On the display form, **add** is the default operation, but you must supply the appropriate service information with the additional arguments.

► *-service string*

Specifies the name of the service. Service names for login and incoming slave services are defined in both the `services` and `telnet-service` database files (for example, `m1slave1` may be the first incoming slave service defined on `#m1`). The service name can be up to 16 characters long.

► `-device_prefix string`

Specifies the device prefix or device name associated with the service. To create a clonable device entry in the `devices.tin` file to accommodate the service, specify the entire device name of the clonable device. For example, to create the clonable device named `tli_nplogm1` to accommodate a nonprivileged login service, specify the entire device name, `tli_nplogm1`.

To create a set of device entries to accommodate the service, specify the device prefix of the device name used by the set of devices. For example, to create devices named `tli_logm1.1`, `tli_logm1.2`, and `tli_logm1.3` for the login devices associated with the default login service, specify the device prefix `tli_logm1`. The device prefix should be no more than 25 characters long.

The shortest string specified in `-device_prefix` cannot be a subset of any device prefix that is longer. For example, if you specify `stcp1` and `stcp1a` as device prefixes, you will receive the following message.

```
WARNING... stcp1 is subset of stcp1a...
Choose another device prefix name or result is
unpredictable
```

You could specify `stcp1` and `stcp2` as device prefixes instead.

To make it easy to identify STCP TELNET devices, consider using `tli` as the first characters of the prefix or device name.

► `-no_keepalive`

**CYCLE**

Prevents STCP from keeping a dormant connection on the socket associated with the service alive. When this option is enabled (the default), the keepalive flag is enabled. (Note that the keepalive flag that is set by the `ifconfig` command **must** be enabled, or this argument will have no effect. See the description of the `ifconfig` command for more information. The template `telnetSERVICE` database file also enables this option for the default TELNET login service. You should enable this option for all of the login and incoming slave services that you define.

► `-linger [number]`

Specifies the number of seconds that STCP will maintain a TELNET connection associated with the specified service after a close operation. The linger value can range from 2 to 32767. If you specify an invalid value when specifying this argument, the command enables the linger option with a value of 15 seconds. The linger option is disabled by default.

► `-no_nodelay`

**CYCLE**

Prevents STCP from sending a smaller-sized packet as soon as it gets it instead of waiting for more data. When disabled, this option causes STCP to hold a smaller-sized packet until it has enough data to transmit. By default, the command

enables this option, thereby enabling STCP to send a smaller-sized packet as soon as it gets it. The template `telnet-service` database file also enables this option for the default TELNET login service. For performance reasons, you should enable this option for all of the login services and incoming slave services that you define.

- ▶ `-description string`  
Provides a brief description of the service (for example, 'slave1 keepalive'). The description can be up to 64 characters long and should be enclosed in apostrophes.
- ▶ `-no_login` CYCLE  
Specifies that the specified service is a slave service, not a login service (the default). If you specify `-no_login`, the `devices.tin` entry for any TELNET devices associated with the service must also specify the value 0 in the `login_slave` field.
- ▶ `-no_privileged` CYCLE  
Specifies that the service enables only nonprivileged users to log in for a TELNET session. By default, both privileged and nonprivileged users and processes can log in for a TELNET session (as defined by the default TELNET login service). If you specify `-no_privileged` for a service, the `devices.tin` entry for any TELNET devices associated with the service must also specify the value 0 in the `priv_terminal` field.
- ▶ `-service_file path_name`  
Specifies the path name of the database file containing the TELNET information. By default, the TELNET database file is  
(master\_disk)>system>stcp>telnet-service.
- ▶ `-services_port number`  
Specifies the port number that should be associated with the service. The default TELNET login service uses port 23. When selecting a port number for a new TELNET service, make sure that the `services` database file does not already associate that port number with an existing service.
- ▶ `-local_ip IP_address`  
Specifies the IPv4 address to which the TELNET service binds, in order to restrict the interface on which a service listens. If you do not specify a value for this argument, the `telnetd` daemon listens on all interfaces.

## Explanation

The `telnet_admin` command automates the process of adding, deleting, or modifying local TELNET services while the STCP stack is up and the `telnetd` daemon process is running. You do not issue this command to configure an outgoing slave service, since outgoing slave services are not included in the local `telnet-service` or `services` database file. When you issue the `telnet_admin`



command to add, delete, or modify a local TELNET service, the command automatically updates the `telnet` service database file as well as the `services` database file. Therefore, the service you manipulate is considered permanent. You can also use the `telnet_admin` command to check the status of a particular TELNET service.

Note that this command enables the `keepalive` and `no-delay` features by default. `Keepalive` and `no-delay` are recommended for all services associated with the `telnetd` daemon process.

When you enter the `telnet_admin` command for an existing service, you **must** include an argument and value that corresponds to each field in the service's entry in the `telnet` service file; otherwise, the command fails. Table 9-23 lists the `telnet` service file fields with their corresponding `telnet_admin` arguments.

**Table 9-23. telnet** service File Fields and Corresponding `telnet_admin` Arguments

<b>telnet</b> service File Fields	Corresponding <code>telnet_admin</code> Arguments
<code>service_name</code>	<code>-service</code>
<code>device_type</code>	(none)
<code>options</code>	<code>-keepalive</code>
	<code>-linger</code>
	<code>-nodelay</code>
<code>description</code>	<code>-description</code>
<code>login_slave</code>	<code>-login</code>
<code>privileged</code>	<code>-privileged</code>
<code>device_prefix</code>	<code>-device_prefix</code>
<code>-local_ip</code>	<code>-local_ip</code>

Although there is only one `telnetd` daemon process, each service must be associated with a unique port number. (The default TELNET service for logins uses port 23.) Therefore, you should review your existing `services` database file and select an unused port number if you plan to add a TELNET service.

Before you issue the command to add a TELNET service, make sure that one or more device entries exist to accommodate the service in the `devices.table` file. You must create the device entries that use the `login` or `incoming slave` service in the `devices.tin` file, re-create the `devices.table` file and place it in the directory `(master_disk)>system>configuration`, and then issue the `configure_devices` command to put the new `devices.table` file into effect.

For login or slave devices, the values in the `login_slave` and `priv_terminal` fields must match the values you supply for the login or slave service with the `telnet_admin` command.

## Examples

This section provides examples of the `telnet_admin` command.

### Example 1

The following example shows how to issue the `telnet_admin` command to get a list of STCP TELNET services.

```
telnet_admin list

Service: telnet    Prefix: tli_logm1 Type: window_term
Option:  keepalive nodelay
Description: Default telnet login
Login: 1  Priv: 1   Port: 23

Service: m1slave1 Prefix: tli_inslv1m1 Type: window_term
Option:  keepalive nodelay
Description: slave1 service
Login: 0  Priv: 1   Port: 950

Service: m1slave2 Prefix: tli_inslv2m1 Type: window_term
Option: keepalive linger=10 nodelay
Description: slave2 linger
Login: 0  Priv: 1   Port: 951

3 Service(s)
```

### Example 2

The following example shows how to issue the `telnet_admin` command to add an incoming slave service.

```
telnet_admin add -service m1slave3 -device_prefix tli_inslv3m1
                -keepalive -linger 15 -nodelay -description 'slave3 linger'
                -no_login -no_privileged -services_port 952
```

This will update the `telnet-service` and `services` database files.

The following sample `telnet-service` database file contains one entry for the default login service and three entries for the incoming slave services (services `m1slave1`, `m1slave2`, and the new `m1slave3`). All services employ the `keepalive` and `no-delay` options; two also employ the `linger` option.

```
telnet    window_term  "keepalive nodelay"  "Default Telnet login"    1 1 tli_logm1

m1slave1  window_term  "keepalive nodelay"  "slave1 service"          0 0 tli_inslv1m1
m1slave2  window_term  "keepalive linger=10 nodelay" "slave2 linger"          0 0 tli_inslv2m1
m1slave3  window_term  "keepalive linger=15 nodelay" "slave3 linger"          0 0 tli_inslv3m1
```

The following sample excerpt from a `services` database file shows how the three slave services correspond to the preceding `telnet-service` database file entries. The `service` field lists a service as it is named in the `service_name` field of the `telnet-service` database file.

```
#
m1slave1      950/tcp
m1slave2      951/tcp
m1slave3      952/tcp
```

The `m1slave3` entry in both the `telnet-service` and `services` database files enables the `telnetd` daemon process to listen at port 952 on the host for incoming slave connection requests. In this example, the host application must already be executing and must have attached to and opened a device with the clonable device name `tli_inslv3m1`. The client must send a connection request to port 952 on the host. When the `telnetd` process receives the connection request, it establishes a TELNET session by attaching the request to an open slave device for the service.

## Related Information

See [“Defining STCP TELNET Devices” on page 4-8](#).

## tftp

### Purpose

The `tftp` command, which implements the standard Trivial File Transfer Protocol (TFTP), enables you to transfer files to and from a local network site without supplying a password. The `tftp` command is a general-user command.

### Display Form

```
----- tftp -----
host: [REDACTED]
transfer_mode: ascii
command: put
from_file: [REDACTED]
-to_file:
-trace: no
```

### Command-Line Form

```
tftp host transfer_mode command from_file
      [-to_file string]
      [-trace]
```

### Arguments

- ▶ *host* **Required**  
Specifies the host name or IPv4 address of the remote host. (Do not specify an IPv6 address; `tftp` does not support IPv6 functionality.)
- ▶ *transfer\_mode* CYCLE **Required**  
Sets the transfer mode for the file. You can specify either `ascii` or `binary`. By default, `tftp` uses the `ascii` transfer mode, which is suitable for text files (ASCII files). Use the `binary` mode to transfer a nontext file (for example, compiled code).

- *command* CYCLE Required

Specifies the operation to be performed, either `put` or `get`. Specify `put` to transfer the specified source file to the remote host; specify `get` to retrieve the specified source file from the remote host. If you do not specify a command, `tftp` uses `put`.
- *from\_file* Required

The path name of the file to be transferred. If the operation is `put`, the source file resides on the local host and will be transferred to the remote host. If the operation is `get`, the source file resides on the remote host and will be transferred to the local host.
- *-to\_file string*

Specifies the name assigned to the transferred file (the copy). You should specify the full path name.

  - If you do not specify a full path name, the default path for a transfer to an STCP host is `(master_disk)>system>stcp>tftp_default`. For security, `modify` permission for this directory does not by default include the user `nobody.nobody`, so attempts to write to this directory will fail. To enable the user `nobody.nobody` to write to this directory, the system administrator explicitly needs to create the directory `((master_disk)>system>stcp>tftp_default)` if it does not already exist and to give `modify` permission to the user `nobody.nobody` for this directory.
  - If you do not specify a file name for the transferred file with the `-to_file` argument, `tftp` simply uses the name of the source file.
- *-trace* CYCLE

Enables tracing, which sends messages concerning the transfer to the console. By default, `tftp` does not enable tracing.

## Explanation

The `tftp` command transfers a file without user authentication (that is, it does not check to see if you are a valid user). It only transfers a file that is readable by anyone to a directory that is writable by anyone.

The `tftp` command can be used to download a boot file to a diskless client host so that the boot file can then be executed.

### NOTE

If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in TFTP services.

## Examples

In the following example, the `tftp` command transfers the file `spec1` from the remote host (172.16.3.35) to the local host.

```
tftp 172.16.3.35 get %sys2#m7-1>Admin>Specs>spec1 -to_file  
+%sys1#m1-1>specs>spec1.new
```

If the transfer is successful, the command shows how much data was transferred. If the transfer is unsuccessful, the command generates an error message in the (master\_disk)>system>stcp>logs>inetd.out file.

## Related Information

See the description of `tftpd` in [Chapter 8](#) for information about the `tftpd` daemon process. In [Chapter 5](#), see “[The `bootptab` File and Boot-Related Information](#)” on [page 5-4](#) for more information about handling boot files, and “[The `hosts.allow` and `hosts.deny` Files and TCP Wrappers](#)” on [page 5-10](#) for information about TCP wrappers functionality.

## tftp6

### Purpose

The `tftp6` command implements an open source version of TFTP (`tftp-hpa`) that supports IPv6 addresses. See open source documentation of `tftp-hpa` for a description of `tftp6` (for example, <http://dev.man-online.org/man1/tftp/>).

## traceroute

### Purpose

The `traceroute` command traces the route that an IP packet follows to the specified network host.

### Display Form

```
----- traceroute -----
host: [REDACTED]
-data_size:      0
-ipv6:           yes
-ipv4:           yes
-max_ttl:        30
-port:
-nqueries:       3
-src_addr:
-tos:            0
-wait_time:      5
-firsthop:       1
-use_icmp6:      no
-use_none:       no
-numeric:        no
-dontroute:      no
-debug:          no
-verbose:        no
```



## Command-Line Form

```
traceroute host
        [-data_size size]
        [-no_ipv6]
        [-no_ipv4]
        [-max_ttl max_ttl]
        [-port port_number]
        [-nqueries number_of_queries]
        [-src_addr source_address]
        [-tos type_of_service]
        [-wait_time wait_time]
        [-firsthop number]
        [-use_icmp6]
        [-use_none]
        [-numeric]
        [-dontroute]
        [-debug]
        [-verbose]
```

## Arguments

- *host* **Required**

The host name or IP address of the remote host.
- -data\_size *size*

Specifies the length of the probe packet.
- -no\_ipv6 CYCLE

Specifies that the *host* argument cannot be an IPv6 address. By default (the value *yes*), the *host* argument can be an IPv6 address.
- -no\_ipv4 CYCLE

Specifies that the *host* argument cannot be an IPv4 address. By default (the value *yes*), the *host* argument can be an IPv4 address.
- -max\_ttl *max\_ttl*

Sets the maximum time-to-live (TTL) used in outgoing probe packets. The TTL refers to the number of hops from one system to another. (A *hop* is the trip a data packet takes from one router or intermediate point to another in the network.) The default is 30 hops.

► `-port port_number`

Specifies a base port number used in probe packets. The default port number is 33434. Starting with `port_number`, the `tracert` command increments the port number in the probe packet each time it sends a message.

The `tracert` command expects the target system to return the message ICMP Port Unreachable. If an application is listening to the port number in the `tracert` message, the target system will not return the ICMP message. Thus, you should specify for `port_number` a base port number that results in unused port numbers on the target system. A large value for `port_number` increases the possibility of unused port numbers.

► `-nqueries number_of_queries`

Sets the number of probe packets per TTL to `number_of_queries`. The default is 3 probe packets.

► `-src_addr source_address`

Specifies the IP address to be used as the source address for outgoing probe packets. The IP address must be a numeric IP address, not a host name.

► `-tos type_of_service`

Sets the type of service in probe packets to the specified value. The default is 0. The value must be a decimal integer in the range 0 to 255. Use this argument to see if different types of service result in different paths.

► `-wait_time wait_time`

Sets the time, in seconds, that `tracert` should wait for a response to a probe packet. The default is 3 seconds.

► `-firsthop number`

Sets the first hop displayed in the command output. By default (the value 1), the command output displays the first hop. The minimum value is 1; the maximum value is 255. This argument is valid only when `host` is an IPv6 address.

► `-use_icmp6`

CYCLE

Enables the `tracert` command to use ICMP6 ECHO packets instead of UDP datagrams. This argument is valid only when `host` is an IPv6 address. If `host` is an IPv4 address, the command ignores the argument and displays a warning message.

► `-use_none`

CYCLE

Enables the `tracert` command to use IPPROTO\_NONE packets instead of UDP datagrams. This argument is valid only when `host` is an IPv6 address.

► `-numeric`

CYCLE

Enables the `tracert` command to display only IP addresses as it traces the route between a module and a specified remote host. When the command displays

only IP addresses, it can ignore host names. By default (the value `no`), the command displays both IP addresses and host names.

- ▶ `-dontroute` CYCLE  
Enables the `tracert` command to use the `SO_DONTROUTE` option on the transmitting socket, and enables the command to transmit on the socket specified by the `-src_addr` argument.
- ▶ `-debug` CYCLE  
Enables debugging mode. By default, debugging mode is suppressed.
- ▶ `-verbose` CYCLE  
Enables the command output to include extra information as well as extra warning and status messages. By default, display of extra information and messages is suppressed.

## Explanation

The `tracert` command traces the route through the Internet between a module and a specified server. It also calculates and displays response time (in milliseconds), which identifies the amount of time it takes for an IP packet to travel the route.

The `tracert` command traces the path to a host by sending bursts of probe packets and then incrementing the TTL on those packets. The first burst of three probe packets has a TTL of one hop, so these packets never get past the first router. Since these packets have exhausted their TTL, the router will send back a Time Exceeded ICMP packet.

By default, three probe packets are sent at each TTL setting, and `tracert` displays the TTL, the address of the router/gateway, and the round-trip time of each probe packet.

The `tracert` command provides you with an approximation of how fast the connection is by measuring the delay between the original packet and the reply. Additional bursts of probe packets are sent with a TTL one hop larger than the previous burst until you reach the limit (by default, 30 hops) or until the host is found and a Port Unreachable ICMP packet is returned. If the probe-packet answers come from different routers/gateways, the address of each responding system is displayed. If no response occurs within a given time-out interval, `tracert` displays an asterisk (\*) for that probe packet.

The `tracert` command is useful for locating possible trouble spots on large and complex networks that are connected together with routers. If your module's response time is slow, you should consider issuing the `tracert` command to see what kind of response the server in question is having.

Before you issue the `tracert` command, it is often helpful to issue the `ping` command to see if a host is present on the network.

For information on `tracert` command output when *host* is an IPv6 address, see <http://www.freebsd.org/cgi/man.cgi?query=tracert6&sektion=8>.

**Table 9-24** lists and describes message characters that can appear in `tracert` output for IPv6 addresses.

**Table 9-24. Message Characters in `tracert` Command Output for IPv6 Addresses**

Character(s)	Description
!N	Destination Unreachable — No Route to Host.
!P	Destination Unreachable — Administratively Prohibited.
!S	Destination Unreachable — Not a Neighbor.
!A	Destination Unreachable — Address Unreachable.
!	The hop limit is $\leq 1$ with a port-unreachable message, which indicates that the packet arrived at the destination, but the reply had a hop limit that was just large enough to allow it to return to the source of <code>tracert</code> .

## Examples

The following example shows the use of the `tracert` command.

```
tracert 192.168.164.16
tracert to 192.168.164.16 (192.168.164.16), 30 hops max, 40 byte packets
 1  router1.stratus.com (192.168.76.1)  69 ms  4 ms  3 ms
 2  router2.stratus.com (192.168.79.1)   4 ms  6 ms  5 ms
 3  172.20.11.49 (172.20.11.49)  28 ms  32 ms  29 ms
 4  172.20.12.4 (172.20.12.4)   30 ms  32 ms  31 ms
 5  172.20.11.1 (172.20.11.1)   98 ms 100 ms  99 ms
 6  172.16.176.21 (172.16.176.21) 224 ms 177 ms 178 ms
 7  172.16.173.18 (172.16.173.18) 190 ms 191 ms 191 ms
 8  172.16.70.18 (172.16.70.18) 188 ms 190 ms 191 ms
 9  172.16.73.98 (172.16.73.98) 199 ms 199 ms 199 ms
10  router3.stratus.com (192.168.164.16) 208 ms 212 ms 211 ms
```

The command output displays only the IP addresses of the 172.20 and 172.16 routers because the `tracert` command could not resolve the host name.

The following two examples show the use of the `tracert` command with an IPv6 address. In the first IPv6 address example, the characters `!N` indicate that no route exists to the host:

```
tracert 2607:f8b0:4006:809::1007
tracert6 to 2607:f8b0:4006:809::1007 (2607:f8b0:4006:809::1007) from
fd73:738c:286:165:200:a8ff:fe48:b2ac, 30 hops max, 12 byte packets
 1 fd73:738c:286:165:: (fd73:738c:286:165::) 2.699 ms 0.910 ms 0.858 ms
 2 fd73:738c:286:165:: (fd73:738c:286:165::) 0.854 ms !N 0.956 ms !N 0.840 ms !N
```

In this second IPv6 address example, the characters `!A` indicate that the destination address is unreachable:

```
tracert fd73:738c:286:f2:200:a8ff:fe4b:b2a1
tracert6 to fd73:738c:286:f2:200:a8ff:fe4b:b2a1 (fd73:738c:286:f2:200:
a8ff:fe4b:b2a1) from fd73:738c:286:f2:200:a8ff:fe4b:b2ac, 30 hops max, 12
byte packets
 1 fd73:738c:286:f2:200:a8ff:fe4b:b2ac (fd73:738c:286:f2:200:a8ff:fe4b:
b2ac) 2992.860 ms !A 3000.992 ms !A 3001.056 ms !A
```

## Related Information

See descriptions of the [ping](#) and [route](#) commands.



---

# Chapter 10

## Troubleshooting Information

This chapter provides troubleshooting information for STCP. Note that you can also use SDLMUX- and STCP-related requests of the `analyze_system` subsystem when you troubleshoot problems with STCP (see [“SDLMUX Requests of the analyze\\_system Subsystem” on page 3-31](#) and [“STCP Requests of the analyze\\_system Subsystem” on page 6-21](#)).

The following sections identify various problems and offer advice on how to diagnose these problems.

- [“Problem 1: Cannot Reach Hosts on the Local Subnet” on page 10-2](#)
- [“Problem 2: Cannot Reach a Specific Host on the Local Subnet” on page 10-3](#)
- [“Problem 3: Cannot Reach a Host on a Different Subnet” on page 10-4](#)
- [“Problem 4: Outgoing TELNET or FTP Connections Fail” on page 10-5](#)
- [“Problem 5: Incoming TELNET or FTP Connections Fail” on page 10-5](#)
- [“Problem 6: Module Cannot Route Packets” on page 10-6](#)
- [“Problem 7: No Communication Using a PCI Adapter” on page 10-7](#)
- [“Problem 8: Communication Problems Associated with a Large Number of Sockets” on page 10-7](#)

As a general rule, you should perform the following tasks to verify that the various components of STCP are operational. Performing these tasks will help you find (or rule out) the most common configuration problems.

- Issue the OpenVOS `list_kernel_programs` command to verify that the STCP protocol stack is up. (If an error occurred during the loading of the STCP protocol stack, an error should have been generated by the `start_stcp.cm` command macro and sent to the console. System-related errors generally appear in the `syserr_log.date` file.) You should see the STCP STREAMS driver `ip.cp.pm` listed in the output. (See [“Verifying the Drivers” on page 6-15](#) for sample output that lists the loaded STCP STREAMS drivers.) In the output, you should also see the related lower-level driver `sdlmux.cp.pm` and the appropriate device drivers (for example, `igb.cp.pm`).
- Issue the STCP `netstat` command with the `-all_sockets` argument to verify that the appropriate daemon processes are running and listening (for example,

`telnetd`, `ftpd`, and `inetd`). (See the Format 1 `netstat` examples in [Chapter 9](#) for sample output.)

- Issue the `list_library_paths` (or `list_default_library_paths`) and specify `command` for the library name to check that you have defined the STCP command library for the module.
- Issue the following `list_users` command to determine which daemon processes are running.

```
list_users *.* -process *d -interval -system -admin
```

- Issue the `ifconfig` command with no arguments to display a list of the network interfaces configured for STCP.
- Issue the `ping` command to verify that the local host can reach another host.
- Check for error messages in the log files generated by `start_stcp.cm`, `start_xinetd.cm` (or `start_inetd.cm`), and/or the daemon processes such as `xinetd` (or `inetd`), `ftpd`, and `telnetd`. In addition, search the `module_start_up.out` file (in `>Overseer`) for `stcp` and look for error messages.
- Issue the OpenVOS `list_devices` command with one of the following arguments.
  - Use the argument `-type streams` to list the STREAMS devices configured on the module.
  - Use the argument `-type streams_pci` to list the ports of Ethernet PCI adapters.

The command output should include STCP-related devices as well as the STCP network interfaces.

- Issue the `route` command and specify the `print` subcommand to check your configuration of STCP routes (for example, to confirm a default router/gateway route or verify the subnet masks for the routes in a VLSN configuration). To ensure that the routes are established each time STCP is started, issue your `route` commands from the `start_stcp.cm` command macro.

## Problem 1: Cannot Reach Hosts on the Local Subnet

One problem that you may encounter is the inability to reach any of the hosts on the local subnet. You may detect this problem when you try to issue STCP `ping` commands to a number of hosts on the local subnet or LAN segment.

If you cannot reach hosts on the local subnet, perform the following checks.

- Verify that the STCP network interface is up (issue the `ifconfig` command with the device name of the network interface). If the state is not up and operational, fix



the configuration. Check the software configuration, such as the `ifconfig` command line in the `start_stcp.cm` command macro, as well as the hardware configuration.

- If the network interface is up and operational, check the IP address, the subnet mask, and the broadcast information (the broadcast address and broadcast type). If any of this information is incorrect, correct the error.
- If you are using stateless autoconfiguration for IPv6, make sure it is configured correctly in the router and that `ifconfig` displays the `use_router_adv nd6` option.
- Issue the `ping` command to ping the loopback interface (`127.0.0.1`). If the command fails, check the configuration of the loopback interface. For example, check the `module_start_up.out` file for errors from the `start_stcp_stack` command.
- Issue the `ping` command using the IP address of a local host. (If this works when a host name fails, the problem is name resolution.) As a check, refer to the information in your STCP `resolv.conf` and/or `hosts` database file. If the command fails, see if other hosts are successfully using the network. If not, issue the `netstat` command with the `-interface` argument and look for errors (bad checksums and so on).
- If other hosts are able to use the network, see if the other host can ping your local module. Also check for CRC or alignment errors, and for late collisions or excessive collisions. These errors indicate a link-layer problem; for example, the interface is configured for full duplex but the switch is set to half duplex.

If the other host cannot ping your local module, examine the ICMP echo counts to check the IP layer, the ARP cache for Ethernet connectivity (issue the `arp` command with the `-all` argument), and the hardware, such as the transceiver and cables.

- Issue the `tcpdump` command to display information about the incoming packets and the broadcast/multicast packets. If there are no incoming packets, there is probably a problem with the connection. If there are broadcast/multicast packets but no unicast packets, the connection is probably incorrect (that is, it is not on the correct subnet).

## Problem 2: Cannot Reach a Specific Host on the Local Subnet

Sometimes, you may be able to reach a number of hosts on the local subnet or LAN segment, but you cannot reach a specific host. You may detect this problem when you try to issue a `ping` command to a specific host on the local subnet or LAN segment.

If you cannot reach a specific host on the local subnet, perform the following checks.

- See if other hosts can reach the specific host; if they cannot, the problem is with that host, so you need to fix it. (Check its hardware and software configuration.)
- If other hosts can reach that host, issue the `ping` command using the IP address of the specific host. If the IP address works, check your `hosts` database file and/or the DNS server information to verify host name resolution.
- If using the IP address of the host does not work, issue the `arp` command with the `-all` argument to see the ARP cache and to verify connectivity. If the ARP cache is incorrect, fix the cache. For example, two hosts may have the same IP address, a board may have been changed, or an ARP entry may have been added manually. In this case, check the MAC address, then delete the incorrect ARP entry. Issue the `ping` command again to generate a new ARP entry.
- If the ARP cache is correct, see if the specific host can ping you. If not, check its echo counts, ARP cache, and all connecting hardware.
- Check that the subnet mask is correct on both the local and remote hosts. If the subnet mask is wrong, you can correct it by issuing the `ifconfig` command with the `-delete` and `-alias` arguments, and then issuing the `ifconfig` command again with the `-add` and `-alias` arguments.

## Problem 3: Cannot Reach a Host on a Different Subnet

If you can ping hosts on your local subnet but not another subnet, a routing problem probably exists, either with the physical configuration (no connection exists to that subnet through a router/gateway, or the router/gateway is out of service), or with an explicitly configured route (configured with the `route` command).

If you cannot reach a remote host, perform the following checks.

- Issue the `route` command with the `print` subcommand to display the routing table. If the routes appear to be incorrect, correct them.
- Check the routes on the remote host. Routes back from the remote host must also exist. If those routes appear to be incorrect, correct them.
- If the subnet mask is correct, see if the router can ping your local module. If not, check the IP address of your module on the router and the ARP cache entry.
- If the router can ping you, check the routes again.

If other hosts on the local subnet cannot reach the remote host using the same router, the problem is most likely with the router (or with the remote host). For example, the router might be configured to drop pings. If other hosts can reach the remote host using that router, try to ping the remote host using the IP address. If that works, the problem is with the information in the `hosts` database file and/or the DNS server (name resolution).

## Problem 4: Outgoing TELNET or FTP Connections Fail

If TELNET or FTP connection requests issued from the STCP module fail, perform the following checks.

- Determine whether the IP address of the target remote host works versus the host name. The problem could be name resolution.
- Verify that the remote host is operational by determining whether other hosts can reach the same host.
- Verify that the remote host is running a TELNET or FTP server.
- Verify that the connection request is made using the TELNET or FTP command from the STCP command library.

## Problem 5: Incoming TELNET or FTP Connections Fail

If incoming TELNET connections fail (but FTP and [ping](#) work), perform the following checks.

- Verify that the STCP TELNET server (`telnetd`) is running by issuing the `netstat` command with the `-all_sockets` argument.
- Verify that the various TELNET STREAMS components are loaded (using the OpenVOS `list_kernel_programs` command).
- Check all of your TELNET device entries as well as the TELNET service entries in the `telnet-service` and `services` database files.
- If a client cannot establish a TELNET or FTP connection, or if a client cannot establish a connection for one of the services configured with the `tcpd` daemon in the `inetd.conf` file (BOOTP and/or TFTP), check the log file `tcpddeny` for a message stating that the client has been denied a connection because it is not listed in the `hosts.allow` file. You should also check the `syserr_log.date` file for messages indicating syntax errors in the `hosts.allow` or `hosts.deny` files.

If incoming FTP connections fail (but TELNET and [ping](#) work), issue a `list_users` or `netstat -all_sockets` command to verify that the STCP FTP daemon process (`ftpd`) is running on the module.

You should also check the appropriate log files for error messages. Check the system error log file (`syserr_log.date`) as well as the STCP log files. Check `telnetd.out` for TELNET error messages. For FTP error messages, check `ftpd.out` and the `ftpd` child process log files (for example, `ftpc.99-07-04_16:05:20_14860.out`).

## Problem 6: Module Cannot Route Packets

If the module is not able to route packets and the `ospfd` daemon process is running, be aware that it updates the routing table only when it learns new routes.

Examine the `syserr_log.date` file for a `route -flush` command and OSPF route entries. If the routing table has been flushed after the OSPF entries were made in the routing table, stop and restart the `ospfd` daemon process to refresh the routing table with OSPF entries.

Debug logs on the local OpenVOS module provide information you can use to troubleshoot the `ospfd` daemon process on the module. The debug logs are in the `(master_disk)>system>stcp>logs` directory and consist of the following:

- `ospfd.log` contains error, state transition, and summary information about input and output activity.
- `ospfd.io` contains everything in the `ospfd.log` file, event reference numbers, any information returned in response to `omon` subcommands, and debugging information. [Appendix E](#) describes the error messages in this file.
- `ospfd.pkt` contains information about all packets sent and received and event reference numbers.

To enable logging on a local or remote OpenVOS module, start an `omon` command session and do any of the following:

- To start debug logging and event logging on the module, issue the `event_logging` subcommand.
- To start debug logging and packet logging on the module, issue the `pkt_logging` subcommand.
- To start debug logging and logging of specific events or packets on the module, issue the `flag` subcommand. Specify the type of event or packet with the `value` argument; [Table 9-20](#) lists the valid values for the argument. For more information, see the description of the `omon` command in [Chapter 9](#).

You can also enable event logging on the local module by issuing the `ospfd` command with the `-form` argument; then, in the display form, next to `event_log`, select `yes`. The `ospfd` daemon displays messages when Open Shortest Path First (OSPF) events occur and logs the messages in the `ospfd.log` file. To stop event logging on the local module, start an `omon` session and issue a `no_event_logging` subcommand, specifying the local router. For more information, see the description of the `ospfd` command in [Chapter 8](#) and of the `omon` command in [Chapter 9](#).

You can obtain OSPF information from `list_errors`, an `omon` subcommand. [Appendix D](#) describes the messages returned by this command.

You can issue the `route` command, specifying `monitor` for the `command` argument to track all changes to the routing table, regardless of the source of the changes.

## Problem 7: No Communication Using a PCI Adapter

If your system cannot communicate with the network using a particular Ethernet PCI adapter, check that the Ethernet PCI adapter cable is properly connected. For information about cables for Ethernet PCI adapters, see the description of the PCI adapter in the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

You should also check the status LEDs of the CPU-I/O enclosure and the Ethernet PCI adapters. The CPU-I/O enclosure contains one pair of LEDs for CPU-I/O enclosure status and one LED for each PCI slot. For information about these LEDs, see the operation and maintenance guide for your system, as listed in [Related Manuals](#). For information about LEDs of Ethernet PCI adapters, see the description of the PCI adapter in the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

## Problem 8: Communication Problems Associated with a Large Number of Sockets

Communication problems can occur when using a large number of sockets. If a socket call fails, you may need to change one or more of the following values:

### NOTICE

Limits for these parameters are high. Before you increase the parameter values, you must study system memory usage to ensure that the operating system does not run out of memory.

- The number of local devices on the module—Change this value using the `-max_local_devices_per_module` argument of the `set_tuning_parameters` command. For a description of the `set_tuning_parameters` command, see *OpenVOS System Administration: Administering and Customizing a System* (R281).
- The clone limit on the STCP driver—Change this value using the `clone_limit_32` field of the `devices.tin` file entry for the STCP driver. For information on the STCP driver, see [“Defining the STCP Driver” on page 4-2](#).
- The TCP transmission control block (TCB) limit—Change this value using the `access_info_monitor` command.

You may need to increase STREAMS memory allocation. To do so, use the following `analyze_system` script:

```
&attach_input
analyze_system
set_streams_param sys_denominator 32
set_streams_param sys_numerator 7
quit
&detach_input
```

OpenVOS may not have enough ports for an application to bind its outgoing sockets to (that is, the application has more outgoing sockets than OpenVOS has ports). In this case, you may need to add the `_TCP_SHARED_DYNAMIC_PORT` socket option to the application. For information, see the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420).

Connections may fail after STCP has established thousands of connections to a subnet. If so, take the following actions:

- Do not change the default configuration for 10Gb Ethernet PCI adapters.
- Failure of connections to a local subnet—A switch or router may be attempting to optimize ARP traffic, but failing because it cannot support large numbers of connections. In this case, configure the switch for more connections. If, however, the switch cannot handle enough connections, check the switch for options such as **Local Proxy ARP** and turn off such options.
- Failure of connections to a non-local subnet—The router's ARP table may be full. In this case, configure a larger ARP table in the router or obtain another router. You can also check the switch for options such as **Proxy ARP** and turn off such options.

---

# Appendix A

## TCP/IP Networking Basics

Before you configure your STCP network, you should be familiar with the networking basics that are discussed in the following sections.

- [“TCP/IP Networks, Subnetworks, Hosts, and Routers/Gateways” on page A-1](#)
- [“Naming Conventions” on page A-3](#)
- [“IP Multicast Transmission Service” on page A-7](#)
- [“Network Interfaces for STCP” on page A-9](#)
- [“Specifying a Network Mask to Identify Subnets” on page A-13](#)
- [“STCP Routes” on page A-18](#)
- [“OSPF Routing” on page A-19](#)
- [“STCP Options” on page A-21](#)
- [“Obtaining IP Addresses and Domain Names” on page A-22](#)
- [“Domain Name Service” on page A-23](#)

### TCP/IP Networks, Subnetworks, Hosts, and Routers/Gateways

A TCP/IP *network* represents the collection of computers (*hosts*) and other devices that communicate using the TCP/IP protocol family. A TCP/IP *subnetwork* (also referred to as a subnet) represents a subsection of a TCP/IP network where all hosts can communicate directly with each other (that is, without a need for routing). Using subnets enables you to customize and expand your LAN configuration. For example, your organization may use a main network that relies on several subnets to accommodate the communications needs and growth of different working groups.

A designated machine called a *router* enables the hosts on separate networks and/or on separate subnets to communicate. (Because routers were called *gateways* in older terminology, this manual uses the term *router/gateway*.) A router/gateway functions as an IP router, and therefore handles the routing of packets from one network to another network and/or from one subnet to another subnet. Although routers/gateways can perform additional protocol-related functions, for the purposes of this manual, a router/gateway is equivalent to an IP router. A simple IP router will perform all of the required functions described in this manual.

Figure A-1 shows a sample network configuration with two main networks, Corporate Engineering and Corporate Administration. The networks are connected using one router/gateway; each of the three subnets in Corporate Engineering (Software, Hardware, and Test) uses a router/gateway to communicate with its main network. The configuration also shows one host, Stratus module #m1.

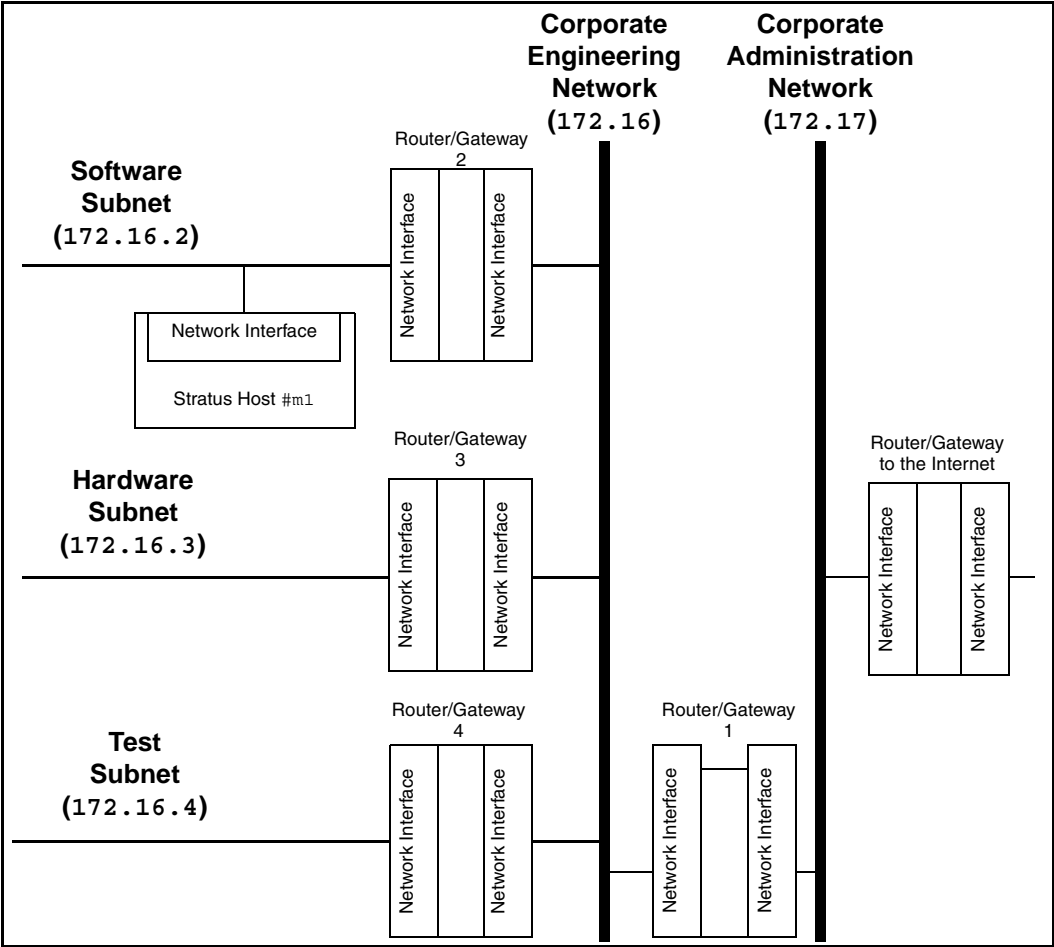


Figure A-1. Sample Network Configuration

NOTE \_\_\_\_\_

The sample network numbers in this configuration (for example, 172.16 and 172.17) are shown only to identify the different components in the sample



configuration. These numbers are not available for use in any way.

As indicated in [Figure A-1](#), each host provides at least one network interface that enables it to communicate over a particular network or subnet. (A network interface is associated with a particular LAN architecture and creates the physical connection to the network or subnet.) A router/gateway provides at least two network interfaces, one connected to each network or subnet.

To enable a host, such as a Stratus module, to communicate with multiple networks or subnets **without** going through a router/gateway, configure that host to provide multiple network interfaces, where each network interface connects directly to a specific network or subnet.

## Naming Conventions

To identify the networks, subnets, and hosts in your network, observe the following naming conventions.

- network numbers
- Internet Protocol (IP) addresses
- host names

Each network you create should be assigned an appropriate network number by an appropriate organization. (See [“Obtaining IP Addresses and Domain Names” on page A-22](#) for information about obtaining IP addresses.) You should obtain a network number for your LAN even if you do not plan to join the *Internet*, the wide area network (WAN) designed to use TCP/IP for data exchange.

The network number determines the format of IP addresses, the addresses that identify all hosts on the network (including those on any subnets). IP addresses are commonly referred to as Internet addresses. In any TCP/IP-based network, you must use IP addresses to define the connection of a host computer or other device to a specific network, and optionally to a subnet of that network. IPv4 addresses use a standard dot-notation form of 32 bits divided into four 8-bit fields, or octets. Each octet can range from 0 to 255. IPv6 addresses are 128 bits long and are divided into 16 bit groups for display purposes. In either case, an IP address consists of the assigned network number, a subnet number that you select (if applicable), and a host number that you select.

In modern networks, all three of these numbers can be an arbitrary number of bits in length. The host number for IPv4 is often 8 bits (for historical reasons), though this is not a requirement. The host number for IPv6 is almost always 64 bits and may be assigned manually or automatically by software based on the MAC address of the interface. IPv4 network numbers assigned by the various authorities are 8 bits or more

in length (though these numbers may be difficult to obtain). IPv6 network numbers are never longer than 48 bits. Most IPv6 network numbers are 48 bits leaving 16 bits to create one or more layers of subnets.

## NOTES

---

1. The length of the three numbers in the IP address (the assigned network number, the subnet number, and the host number) must all add up to the address length.
2. The length of the network number is arbitrary, but the assigning authority, rather than the end user, selects the network number.

Using the sample network configuration in [Figure A-1](#), networks 172.16 and 172.17 have 16-bit network numbers that use two octets to identify the network and have two octets available to identify subnets and hosts. In the figure, 172.16.2, 172.16.3, and 172.16.4 are subnets of the network 172.16.

To identify the subnets in [Figure A-1](#), you build on the format 172.16.xxx.xxx and use the third octet of the IP address to identify a subnet. For example, you identify the Software group subnet as 172.16.2.xxx, the Hardware group subnet as 172.16.3.xxx, and the Test group subnet as 172.16.4.xxx. The fourth octet is typically reserved to identify an individual host's connection to the subnet. For STCP, you should define all IPv4 subnets as well as main networks in the `networks` database file, and you must define all hosts in either the `hosts` database file or a DNS server. (See [Chapter 5](#) for descriptions of the STCP database files.) In addition, you must define each logical network interface with the `ifconfig` command.

In [Figure A-1](#), you might assign the first network interface on host #m1 on subnet 172.16.2 the following IP address.

```
172.16.2.14
```

For more information about the Internet addressing scheme, see <http://www.iana.org/>, which is the Web site of the Internet Assigned Numbers Authority (IANA).

In addition to IP addresses, hosts and routers/gateways are also identified by *host names*. Host names simplify the identification of individual hosts for users. Most applications that accept an IP address as an argument also accept a host name. As defined by RFC 952 and RFC 1123, a host name is a text string of up to 255 characters. The name can contain any letters in the alphabet, the numbers 0 through 9, the period (.), and the hyphen (-). The first character can be either alphabetic or numeric. A host name maps to an IP address. Note that host names are case-insensitive.

To map a host name into an IP address, the Internet relies on a database system called the Domain Name Service (DNS). Within this system, a simple host name (such as `m3-1`) is expanded to include the names of the domains, or zones, to which the host belongs. Domain names place the host within the following:

- a group of machines (a local domain)
- a company or organization encompassing the group of machines (an upper domain)
- a top-level Internet domain

The organization is similar to specifying a street, city, and state in a mailing address. The Internet uses a hierarchy of domains; when you register your network with the NIC, you are assigned to a top-level domain (for example, `com`, the domain name that represents commercial businesses). You can name your upper and local administrative domains (for example, the company and group names), and you should register the upper domain name with the higher-level domain. You should use periods to separate the domain names, and you typically omit the trailing period that represents the root Internet domain. A complete symbolic name would begin with the simple host name followed by a local domain name, an upper domain name (such as a company domain name), and the top-level domain name. For example:

```
m1-1.sw.corp.com
```

In the preceding example, `m1-1` is the simple host name used to identify module #`m1` and network interface number `1`, `sw` is the local domain name, `corp` is the upper domain name, and `com` is the top-level domain name. The entire name acts as a synonym for the IP address of the host.

TCP/IP network configurations can use DNS servers to perform the lookup capabilities (for example, locating the IP address associated with a given host name). If your network configuration includes DNS servers, you should identify the DNS servers in the `resolv.conf` database file. You can also map the host names to IP addresses in the `hosts` database file. (For more information about these database files, see [Chapter 5](#).)

[Figure A-2](#) shows the assigned host names and IP addresses for the hosts and other devices that are part of the sample network configuration in [Figure A-1](#).

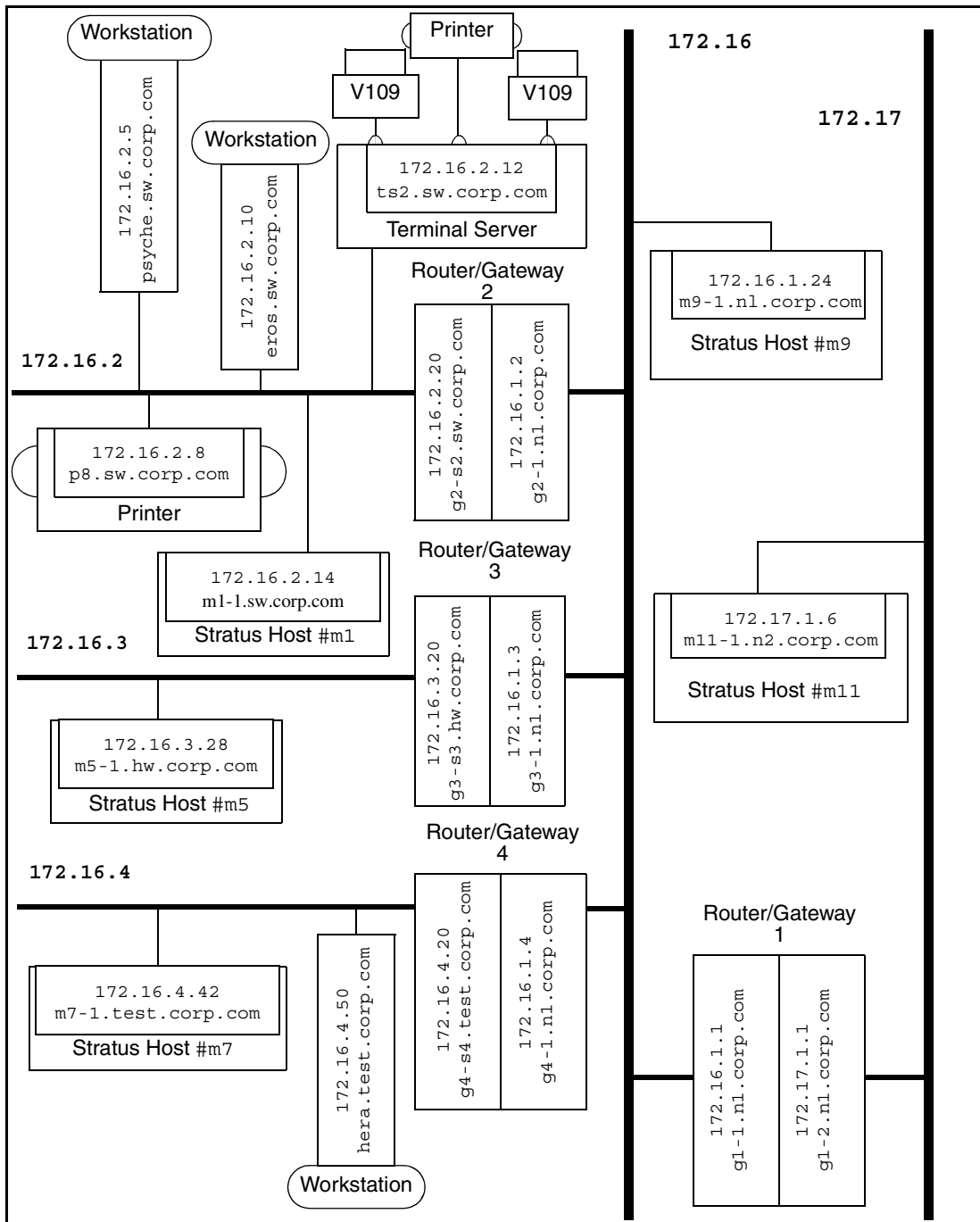


Figure A-2. Configuration with IP Addresses and Host Names

## IP Multicast Transmission Service

Most network users are familiar with point-to-point, or *unicast* service. In this type of transmission service, a sender transmits data packets to a single specific destination; that is, one interface sends the information, and only one receives it. This is the standard form of transmission service provided by networking protocols such as TCP.

The IP multicast transmission service is more suitable for data that is intended for a potentially wide audience. The *IP multicast* transmission service allows users to send data packets to any receiver that is “listening,” rather than to a single specific destination. Data packets are not sent individually to each receiver; instead, the sender transmits only one packet to a group address to which all receivers are listening. This is accomplished by a router running an IP multicast routing protocol. The sender must know in advance what the group addresses are.

OpenVOS Release 18.0.0 and later supports IP3 multicast version v4, as described in RFC 1112 and RFC 3376. IPv4 multicast level version 3 allows you to receive IP multicast data packets. OpenVOS Release 18.0.0 and later also supports IPv6 multicast version 2 (which has the same functionality as IPv4 version 3).

IP multicast can run only with UDP and RAW sockets. Therefore, an IP multicast data packet is not guaranteed to reach all members of the group.

No true set-up processing exists for sending IP multicast. Once an address is determined, the sender simply transmits packets to that multicast address with routers determining the paths taken by the data.

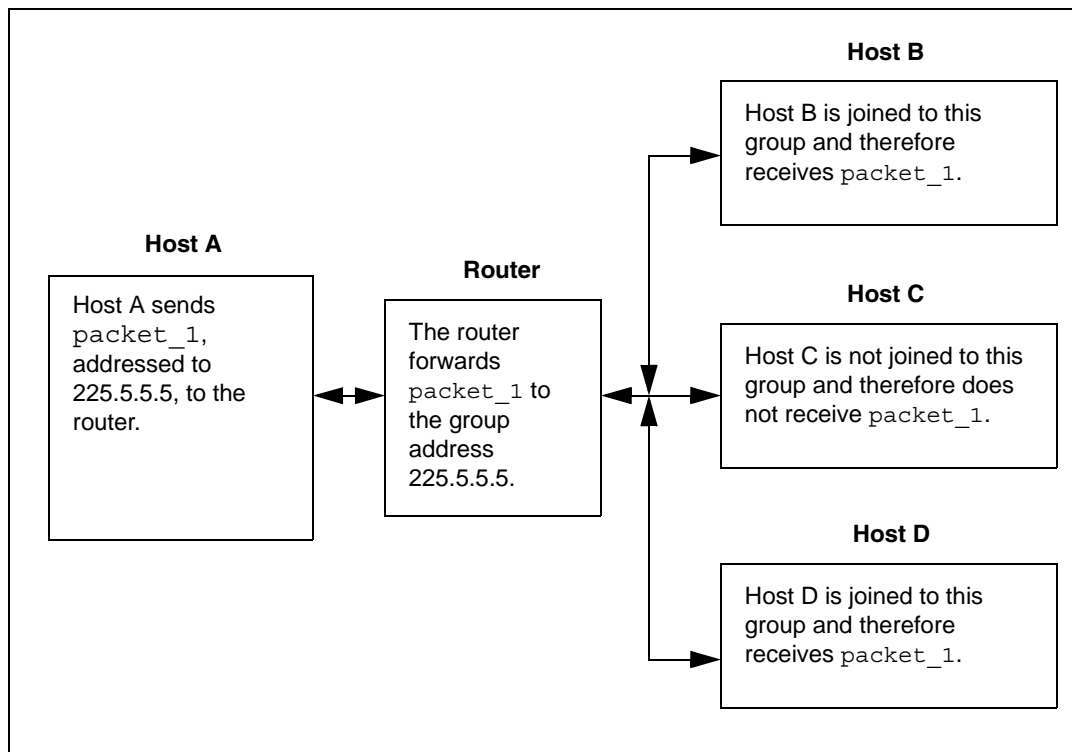
On the receiver side, an application must specify the multicast addresses it wants to receive and, optionally, the sending hosts it wants to receive from. To do that, it uses something like the `setsockopt IP_ADD_MEMBERSHIP` socket option for the IP layer to listen to traffic destined to the desired address. This will register the IP multicast address to the specific interface and will send out the appropriate IGMP (or ICMPv6 MLD) messages to the router (if necessary). Multicast packets are then routed like all other IP packets, with receivers accepting traffic addressed to joined groups in addition to the normal host address.

[Figure A-3](#) provides an example of how an IP multicast packet is sent and received.

### NOTE

---

STCP does not support forwarding of multicast packets, regardless of how the forwarding options are set.



**Figure A-3. Example of Sending and Receiving an IP Multicast Packet**

In [Figure A-3](#), note that in order to **receive** `packet_1`, each host must have joined the group 225.5.5.5. Also, Host A does not need to be a member of group 225.5.5.5 in order to **send** `packet_1`. Note also that configurations do not require a router if they are on the same subnet, but do require switches that support multicast.

An application that no longer wishes to accept IP multicast data packets must explicitly notify the IP layer, either by specifying the `setsockopt IP_DROP_MEMBERSHIP` socket option, or by closing or unbinding the socket. After the application stops accepting IP multicast data packets, the IP layer discards any subsequent IP multicast data packets sent to that group. For more information about the `IP_DROP_MEMBERSHIP` and `IP_ADD_MEMBERSHIP` socket options or about closing and unbinding sockets, see the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420).

The only difference between an IP multicast data packet and an IP unicast data packet is the presence of a multicast address in the Destination Address field of the IP header. The high-order bits of an IPv4 multicast address are 1110, which means that they

range from 224.0.0.0 to 239.255.255.255. The high-order 8 bits of an IPv6 multicast address are 11111111. For more information about address assignments, consult the Internet Assigned Numbers Authority (IANA) web site ([www.iana.org](http://www.iana.org)). Group membership is determined dynamically using the IGMP (or ICMPv6 MLD).

No applications should use multicast addresses that are reserved by the IANA for specific purposes. For example, do not use 224.0.0.1 (representing the ALL HOST group) and 224.0.0.2 (representing the ALL ROUTER group). These addresses are special addresses for the IP multicast protocol, and STCP automatically registers these special addresses with each STCP interface. This is true even if the computers are not connected to the Internet, because many multicast addresses are reserved for specific local network functions.

## Network Interfaces for STCP

To run STCP on your module, you must install and configure each network interface that represents the module's physical connection to a main network or subnet. For example, for an Ethernet network configuration, you must define a port on an Ethernet PCI adapter as the module's network interface to the Ethernet-based network or subnet. For a list of the network interfaces that STCP supports on ftServer systems running OpenVOS, see the *Stratus ftServer V Series Systems: PCI Adapter Guide* (R679).

The following sections summarize the procedures involved in configuring network interfaces.

- “[Overview of Procedures for Network Interfaces](#)” on page A-9
- “[Connecting Interfaces to Multiple Networks or Subnets](#)” on page A-10
- “[Connecting Interfaces to the Same Network or Subnet](#)” on page A-12

## Overview of Procedures for Network Interfaces

To configure a network interface for STCP, you do the following:

- Create device entries in the `devices.tin` file based on the protocol stack and partnering status. (After adding the entries, you must update the `devices.table` file and place the table in the directory `(master_disk)>system>configuration.`) Since each type of network interface requires specific information in the device entry, see [Chapter 3](#).
- To configure two physical network interfaces to function as partners (that is, as one logical interface), ensure that you have met the requirements for defining SDLMUX. (SDLMUX is a communications driver that manages the lower-level device drivers and partnered as well as unpartnered network interfaces.) See [Chapter 3](#) for information about configuring partnered network interfaces.

- Ensure that the related device drivers are loaded at each bootload (from the `module_start_up.cm` file). For SDLMUX, the `start_stcp.cm` file must contain the appropriate `dlmux_admin device_name init_sdlmux` command lines, which load SDLMUX and initialize the device. For detailed information about these procedures, see [Chapter 3](#).
- Create entries for each logical interface by issuing an `ifconfig` command from the STCP `start_stcp.cm` command macro. The `ifconfig` command defines a network interface by assigning an IP address to the specified network interface. (See [Chapter 6](#) for information about the `start_stcp.cm` command macro; see [Chapter 9](#) for information about the `ifconfig` command.)

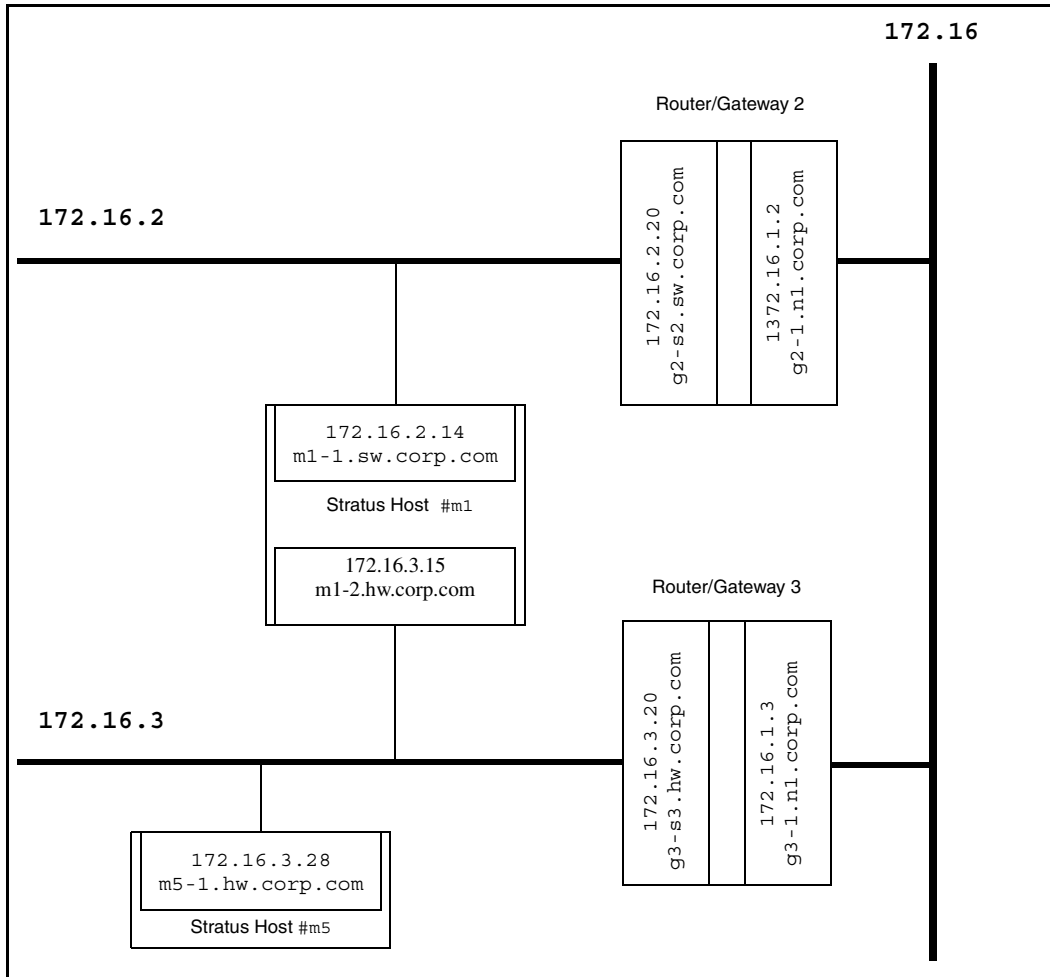
Note that the OpenVOS device name you assign to the network interface in the device entry is the name you must subsequently use to reference the device, and you must precede the device name with the number sign (#). For example, a valid device name specified with an STCP command such as `netstat` is `#enetC.m1.13.2`. Note that if you have configured partnered network interfaces, the device name you specify with the `ifconfig` command is the one that must be used with STCP commands to reference the partners.

## Connecting Interfaces to Multiple Networks or Subnets

In a simple configuration, a Stratus module running STCP provides one network interface that establishes a connection to a particular network or subnet. However, depending on the needs of your own network configuration, you may want the module to create multiple network interfaces. With multiple network interfaces, a module can connect directly to different main networks or different subnets, thereby enabling it to communicate with hosts on the different networks or subnets without going through a local router/gateway. When the IP forwarding feature is enabled, the module can also serve as a router/gateway between the networks or subnets. (By default, the module can act as a router/gateway, which you can verify or change with the `IP_forwarding` command.) The module must provide one network interface for each network or subnet to which you want to connect.

As shown in [Figure A-2](#), a Stratus host (module #m1) connects directly to subnet 172.16.2 using the network interface with IP address 172.16.2.14. It relies on the local router/gateway that provides network interface 172.16.2.20 to communicate with the other 172.16 subnets and the 172.17 network. If #m1 provides the additional network interface 172.16.3.15, as shown in [Figure A-4](#), it can then communicate directly with subnet 172.16.3 without going through the local router/gateway that provides network interface 172.16.3.20. To define this second network interface, you create an additional device entry and issue an `ifconfig` command from the `start_stcp.cm` command macro. Using multiple network interfaces also warrants consideration of routing, which is described later in this appendix.





**Figure A-4. Connecting Network Interfaces to Different Subnets**

## Connecting Interfaces to the Same Network or Subnet

If you configure a Stratus module to provide multiple network interfaces to the same IP network or subnet, you provide multiple paths of communication on the same LAN segment. How the network interfaces connected to the same LAN segment function depends on whether or not they are partnered.

If you create this type of configuration and you configure two same-type network interfaces as partners, SDLMUX ensures hardware failover if the partner handling the communications I/O fails. This preserves the established connections.

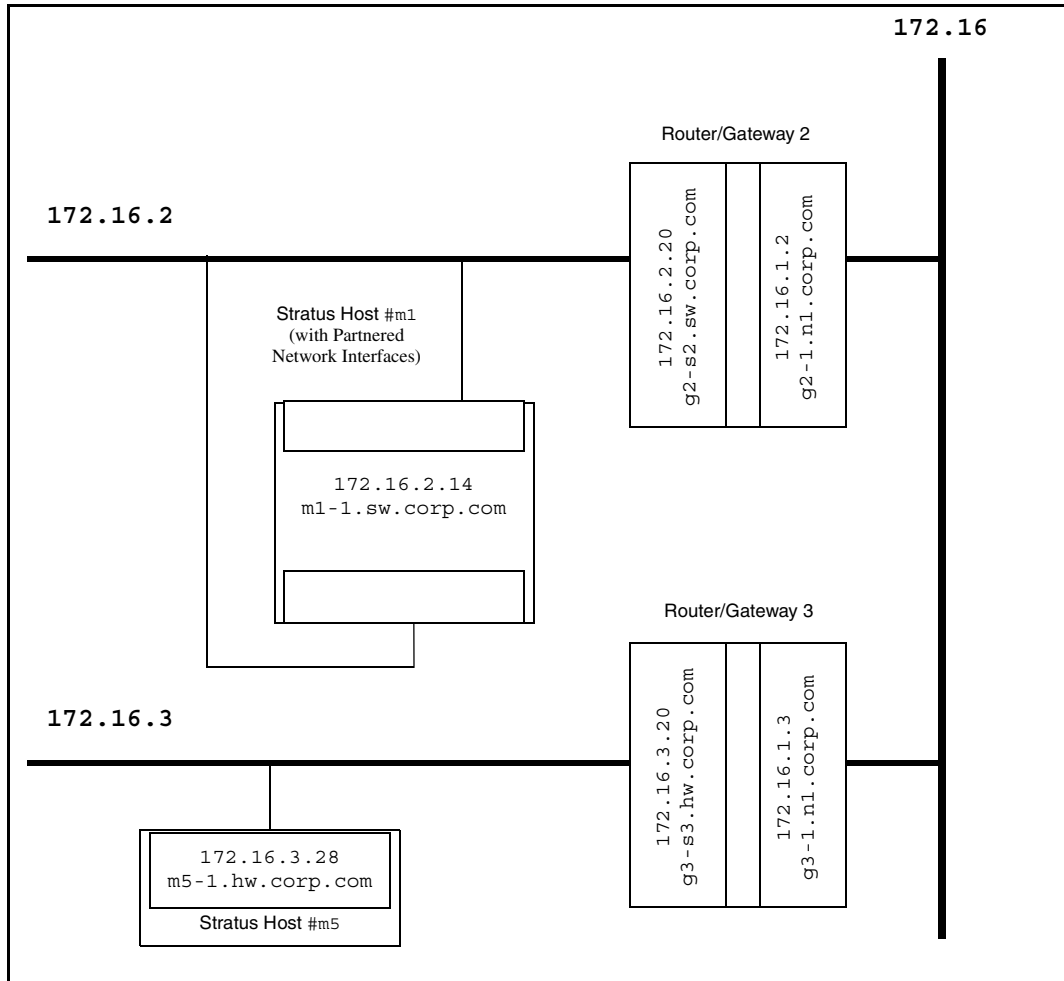
### NOTE

Stratus recommends that you connect only partnered interfaces to the same subnet.

If you do not partner two network interfaces connected to the same LAN segment, this configuration prevents the failure of a single network interface from disabling all subsequent communication with the network, but it does not preserve the established connections by providing automatic hardware failover. (One network interface transmits all outgoing messages; either one can receive incoming messages.) When the network interface handling the communications I/O goes out of service, all established connections to that interface fail once the time-out expires. STCP then requires all subsequent connections to be established using the network interface that is in service. All subsequent connection requests from remote clients **must** be addressed to the operational network interface; if the connection requests are bound for a network interface that is out of service, the connection requests will fail.

Each independent network interface must have its own IP address and must be uniquely defined to STCP. Partnered network interfaces are configured to share one IP address; that is, the IP address of the one partner (the one you configure with the STCP `ifconfig` command) applies to both of the interfaces.

Figure A-5 shows two partnered network interfaces on #m1 that connect to the 172.16.2 subnet. From the perspective of STCP, the IP address 172.16.2.14 is used to refer to **both** partners. (The network interfaces have unique device names and are managed individually by SDLMUX.) If the network interface handling the communications I/O for the subnet goes out of service, STCP starts transmitting over the partner interface that is in service.



**Figure A-5. Configuring Network Interfaces to the Same Subnet**

## Specifying a Network Mask to Identify Subnets

When you configure network interfaces that are directly connected to subnets, you should specify the correct network mask. A *network mask* is a value specifying the number of bits that are used to identify a network number, a subnet number, and the local hosts. All bits used to specify the network ID and subnet ID are set to 1; all of the remaining bits, which are used to specify a host ID, are set to 0. An IPv4 subnet mask can be represented in hexadecimal or decimal and includes four octets. In decimal, these octets are separated by periods (for example, 255.255.255.0).

IPv4 and IPv6 addresses with masks (that is, prefixes) can be formatted as *addr/N*, where *N* is the number of 1 bits in the high part of the mask. So, for example, 255.255.255.0 can be expressed as /24. This formatting is referred to as classless inter-domain routing (CIDR) notation.

The `ifconfig` command enables you to specify a subnet mask (for example, if you are using a subnet mask that deviates from the standard mask for the network class). You can display the subnet mask with either the `ifconfig` command or the `netstat -routing` command. You can use the `route` command to specify a mask for a network route.

The number of bits used for the subnet mask determines how many subnets you can have and how many hosts you can have per subnet. Therefore, the value that you specify for the subnet mask depends on how many subnets and hosts you may need to accommodate in the future. Based on the number of subnets, determine the number of bits in the host ID that are required for the subnet mask, then evaluate the possible number of hosts allowed per subnet. Consider using additional bits in the subnet mask, as described in the next section, if you will need more subnets in the future. (Requiring more subnets means you will then require additional bits from the host ID.) Taking additional bits from the host ID is suitable if you will never need all of the hosts allowed by the remaining bits. The goal is to avoid reassigning IP addresses in the future because doing so is considerable work and your network will be down while you performing the work.

Traditional subnet masks were based on the 8-bit octets in a 32-bit IPv4 address, but this division was dropped long ago due to the scarcity of IP addresses.

## Creating Variable-Length Subnets

STCP supports the use of variable-length subnets (VLSN), which provides flexibility when designing subnets and specifying the associated subnet masks. The 8-bit subnet mask 0xfffff00 (255.255.255.0) may be suitable for many subnet configurations. However, your particular network configuration may warrant a different network mask if you need more than the permitted number of subnets (which means fewer hosts) or if you need more than the permitted number of hosts (which means fewer subnets). With 16 bits available to define the subnet and the host, you may decide to deviate from the 8 bits for subnet and 8 bits for host scheme and select a different distribution.

RFC 1878 addresses VLSN configurations and provides tables that can help you create a VLSN configuration based on the traditional subnetting of the different network classes. It lists the possible subnet masks, the number of subnets and hosts allowed, the possible range of host addresses, and the broadcast addresses. When using these tables, note that variable-length subnetting moves towards a classless network environment, so do not focus on the classifications.

Figure A-6 shows a sample VLSN configuration. In this configuration, the 172.16.4.0 subnet looks like one subnet to the rest of the network, but actually consists of subnets 172.16.4.0, 172.16.4.128, and 172.16.4.192. In a traditional configuration, these subnets would all have unique subnet IDs such as 172.16.4, 172.16.5, and 172.16.6, but variable-length subnetting preserves the range of subnet numbers with fewer hosts per subnet.

In Figure A-6, all network interfaces associated with the 172.16.4.0 subnet must be defined to have the subnet mask 255.255.255.128 (0xfffffff80), which is a 9-bit subnet mask using 9 bits for the subnet ID and 7 bits for the host ID. The network interface 172.16.4.17 on the multihomed host #m4 (which acts as a router) must recognize the subnet mask in order to forward packets correctly. Note that the 172.16.3 and 172.16.4.0 subnets are both part of the 172.16 network, but the 172.16.3 uses the traditional 8-bit subnet mask of 255.255.255.0 (0xfffffff00), which allows 8 bits for the subnet ID and 8 bits for the host ID.

On the 172.16.4.128 and 172.16.4.192 subnets, the subnet mask is 255.255.255.192 (0xffffffc0), which is a 10-bit subnet mask using 10 bits for the subnet ID and 6 bits for the host ID. The address 172.16.4.129 represents the first host (host 1) assigned on the 172.16.4.128 subnet (128 + 1) and 172.16.4.130 represents host 2 (128 + 2). On the 172.16.4.192 subnet, the address 172.16.4.193 represents host 1 on the 172.16.4.192 subnet and 172.16.4.194 represents host 2. (The tables in RFC 1878 can help you determine the valid addresses associated with a given subnet mask.) In Figure A-6, the numbers shown in parentheses indicate the actual host IDs on the 172.16.4.128 and 172.16.4.192 subnets.

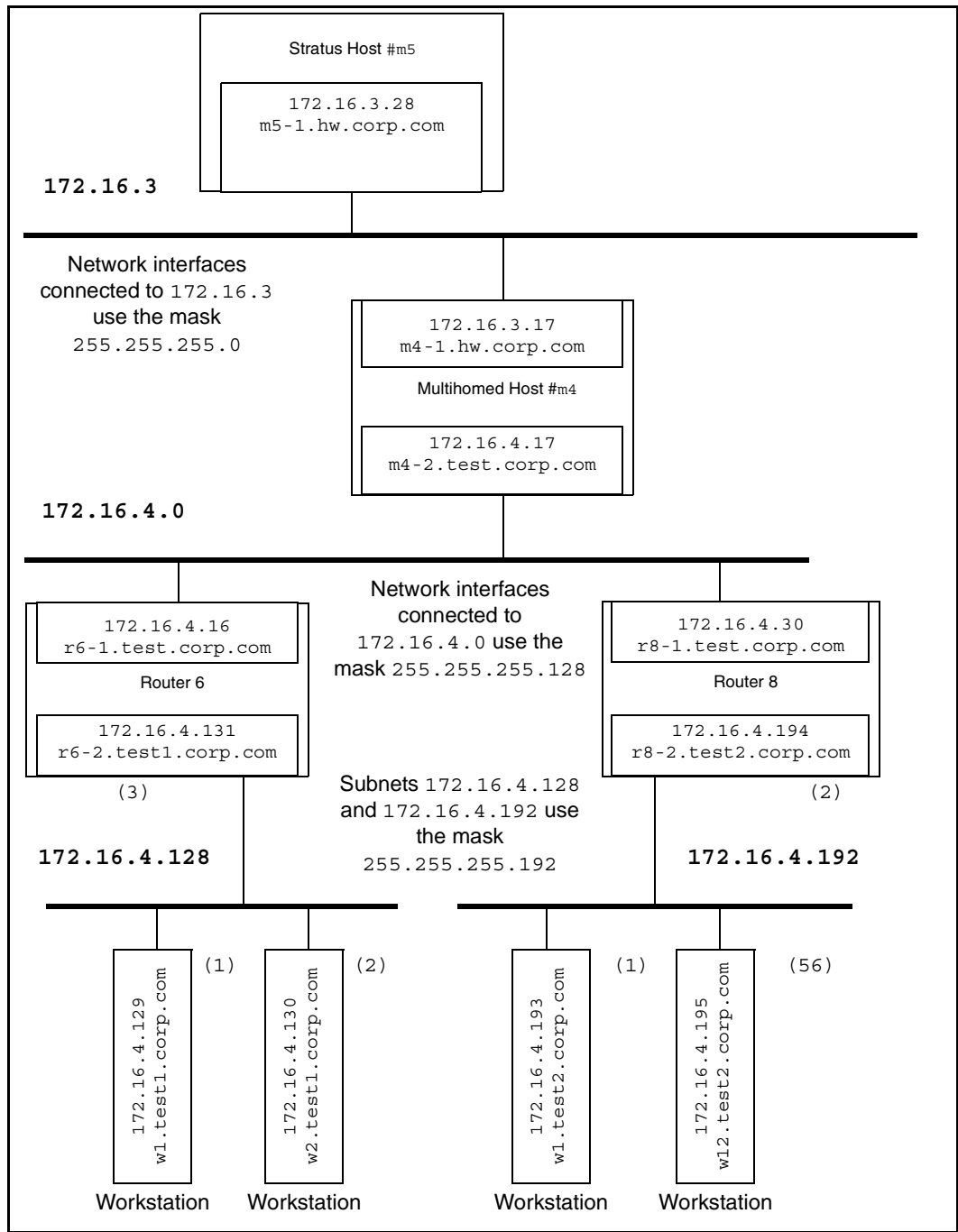
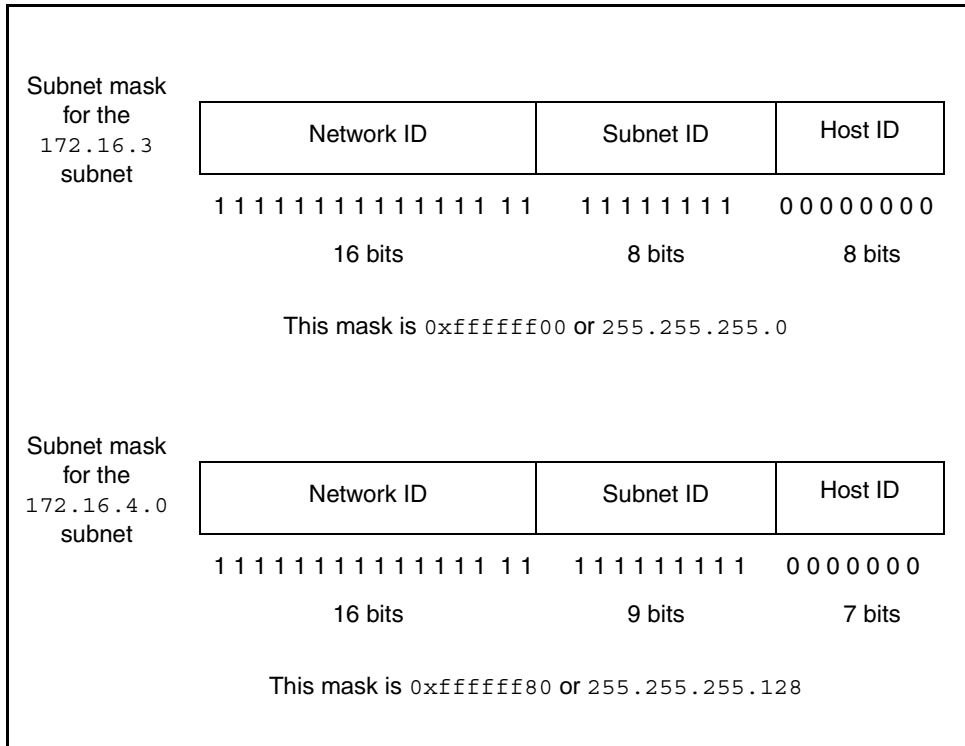


Figure A-6. Sample VLSN Configuration

Figure A-7 shows two different subnet masks used for 172.16.3 and 172.16.4.0 in the 172.16 network.



**Figure A-7. Sample Subnet Masks**

The following example shows the two `ifconfig` commands that you would issue to configure the network interfaces on host #m4 in the sample configuration shown in Figure A-6.

```
ifconfig #enet.m4.4.13 172.16.3.17 -netmask 255.255.255.0 -add
ifconfig #enet.m4.4.12 172.16.4.17 -netmask 255.255.255.128 -add
```

The following example shows the `route` commands that you would issue to create the routing table on host #m4 in Figure A-6.

```
route add 172.16.4.128 172.16.4.16 255.255.255.192
route add 172.16.4.192 172.16.4.30 255.255.255.192
```

## STCP Routes

A *route* is a path over which packets travel to a particular destination. For communication within a particular STCP network (without subnets) or within a particular STCP subnet, you establish a route by configuring the local network interface (using the `ifconfig` command) that is connected directly to that network or subnet. This is a direct implicit route. When you configure an interface using the `ifconfig` command, STCP automatically creates a route for the **local** network/subnetwork, based on the network address and subnet mask of a logical interface. These local routes are part of the routing table.

For STCP communication with hosts and other resources on remote networks or subnets that must be accessed indirectly (that is, through a router/gateway), you must explicitly define routes using the `route` command or use a routing daemon program to maintain the configuration automatically. Note that only one route can be defined to a given destination.

STCP uses the routing information it receives to create and reference a network *routing table*, which is the database of all direct and indirect routes. You can display the routing table by issuing either the `netstat` command with the `-routing` argument or the `route` command with the `print` subcommand. (For descriptions of STCP commands, see [Chapter 9](#).)

STCP recognizes three main types (levels) of routes.

- Routes to hosts (level 1)—These can be thought of as having a netmask of all 1s.
- Routes to [sub]networks (level 2)
- Default route (level 3, through a default router/gateway)—These can be thought of as having a netmask of all 0s.

When STCP references the routing table, it searches **first** for routes to hosts, then routes to networks, then a default route. More specifically, it picks the route with the longest netmask when there are overlaps (overlaps are not allowed with the same netmask). This means that STCP looks for the most specific route (a route to a particular host) first and the least specific route (the default route, which requires only the specification of a local router/gateway to handle routing to remote networks and subnets) last. Routes to hosts are beneficial when you want to handle communication to a particular remote host differently than to other remote hosts. Routes to networks help establish paths to specific networks or subnets and may be helpful when you need to separate the communication to a particular network or subnet and you have multiple routers/gateways that go to the same networks or subnets.

For each network interface that you configure, STCP adds the network interface as an available interface to the LAN. For example, when you add the network interface `172.16.2.14` (using the specified network mask), STCP recognizes the connection to the `172.16.2` subnet and enables the module to communicate with other hosts



whose subnet address is 172.16.2 using the network interface 172.16.2.14 as the source address.

## OSPF Routing

Open Shortest Path First (OSPF) is an interior router/gateway routing discovery protocol that distributes routing information among routers in an *autonomous system* (AS), a collection of hosts and routers that are administered by a single organization. OSPF is a link-state protocol, where each router maintains a *link-state database* (LSD) that describes the topology of the AS and that determines a shortest path based on a large range of metrics, with a cost range from 1 to 65,535. The metrics you use to calculate a cost include line speed. For more information, refer to RFC 1253. (For additional information about OSPF, see RFC 2328 as well as books about Internet Protocol routing in general and about OSPF routing in particular.)

OSPF ensures that if a link or router becomes unavailable, the hosts uses alternate routes.

When the router initializes and broadcasts Hello packets to the subnet at specified intervals, the Hello packets enable neighboring routers to discover one another and build a shared database of routing information. After the exchange of Hello packets, the routers in an area elect a designated router (DR) and a backup designated router.

Each router builds a link-state advertisement (LSA) that describes its own interfaces and reachable neighbors. A router builds the LSD from the collective LSAs from all routers in the AS. All routers on a broadcast subnet keep their LSDs synchronized with that of the DR. Individual LSAs are sent to the DR, and the DR sends summary LSAs to the rest of the routers on the subnet.

OSPF allows networks to be grouped into areas. A module that has multiple interfaces can participate in multiple areas and can act as a border router. Border routers send summary LSAs from one area to another, hiding the details of any one area from the other areas in an AS.

The following sections provide additional information about OSPF:

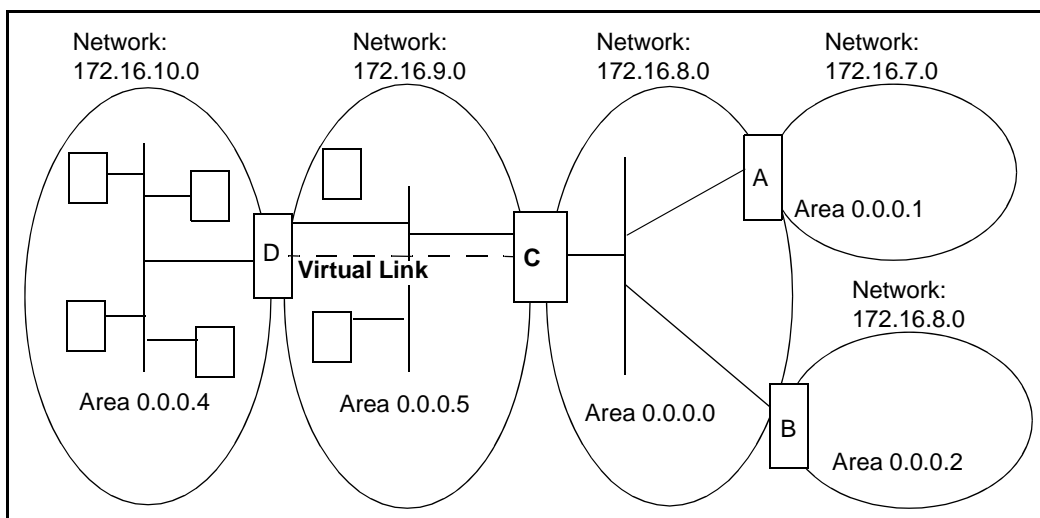
- [“Virtual Links” on page A-20](#)
- [“Boundary Routers” on page A-20](#)
- [“NSSA” on page A-21](#)
- [“Stub Areas” on page A-21](#)
- [“Authentication” on page A-21](#)

## Virtual Links

Each AS has a network that acts as a special area called the OSPF backbone area, which has an area ID of 0.0.0.0. Each area must have a border router that provides an interface to the backbone area. This attachment to the backbone need not be physical. Instead, you can connect an area logically to the backbone through the use of virtual links.

You can create a virtual link between two border routers that have an interface to a common area that is not part of the backbone. The OSPF daemon uses the virtual link to route traffic only within its area.

**Figure A-8** demonstrates a virtual link between router D and C that connects area 0.0.0.4 to the backbone, area 0.0.0.0. routers A, B, and C have physical links to each other, and therefore to the backbone, but no router in area 0.0.0.4 is physically connected to the backbone. Therefore, you must create a virtual link to the backbone between a router in area 0.0.0.4 and router C, which is physically connected to the backbone.



**Figure A-8. Virtual Links**

## Boundary Routers

A boundary router provides the interface between an OSPF AS and an external AS.

Because OpenVOS does not support any other routing protocol, this version of OSPF does not support OpenVOS modules as boundary routers.

## NSSA

This version of OSPF does not support not-so-stubby areas (NSSA).

## Stub Areas

A *stub area* has a single interface to the rest of the AS.

## Authentication

This version of OSPF supports authentication by passwords. All routers in an area must share a common password.

## STCP Options

When you configure STCP, you can specify certain standard options that affect the capabilities of a network interface or the handling of a connection. [Table A-1](#) briefly describes some of the common STCP options. For information about how the options such as keepalive, linger, and no-delay apply to TELNET specifically, see “[The telnet service File and TELNET Information](#)” on page 5-53.

**Table A-1. STCP Options** (Page 1 of 2)

Characteristic	Description
broadcast type	Specifies how STCP applications will send broadcast packets. You specify a broadcast type for a network interface with the <a href="#">ifconfig</a> command. A broadcast type of 0 means that the local host address is all zeros; that is, all bits in the IP address except those comprising the network/subnet number take the value 0. A broadcast type of 1 means that the local host address is all ones (that is, all bits in the IP address except those comprising the network/subnet number take the value 255; for example, 172.16.2.255). STCP applications send broadcasts of the type you specify, but can receive either type. The default broadcast type is 1.
forward broadcast	Specifies whether or not broadcast packets from other networks or subnets can be passed onto the network or subnet associated with the network interface you are configuring. You can enable this option if you have one or more network interfaces that connect to different networks or subnets and the module is able to act as a router/gateway. Turning this option on is dangerous because doing so can create broadcast storms. (By default, the module does not act as a router/gateway, which you can verify or change by issuing the <a href="#">IP_forwarding</a> command, which is described in <a href="#">Chapter 9</a> .)

**Table A-1. STCP Options** (Page 2 of 2)

Characteristic	Description
keepalive	Specifies whether packets are periodically sent to keep idle connections from timing out. An application can set keepalive as a socket option for a connection. For a network interface, you can issue the <code>ifconfig</code> command to enable or disable keepalive as a global flag; by default, the keepalive flag is enabled for a network interface. Note that your application <b>must</b> also set the <code>SO_KEEPALIVE</code> socket option or the keepalive flag will not be enabled. For more information about setting the <code>SO_KEEPALIVE</code> socket option, see the description of the <code>setsockopt</code> function in the <i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420).
linger	When enabled, this option causes STCP to maintain a connection for a specified number of seconds after a close operation. This provides some additional time for pending data to be delivered. When the linger option is disabled, all pending data is discarded when the socket for the connection is closed. To enable the linger option, note that your application <b>must</b> set the <code>SO_LINGER</code> socket option. For more information about setting the <code>SO_LINGER</code> socket option, see the description of the <code>setsockopt</code> function in the <i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420).
no-delay	When enabled, this option causes STCP to send a packet that is smaller than the maximum segment size (MSS) as soon as the application passes it to the STCP driver. When disabled, this option causes STCP to buffer data passed to it by the application until it has MSS bytes of data buffered (then it sends one packet of MSS bytes) or until all data previously sent has been acknowledged (then it sends all the bytes in its buffer in one packet). An application can set the no-delay option as a socket option for a connection. To enable the no-delay option, note that your application <b>must</b> set the <code>SO_NODELAY</code> socket option. For more information about setting the <code>SO_NODELAY</code> socket option, see the description of the <code>setsockopt</code> function in the <i>OpenVOS STREAMS TCP/IP Programmer's Guide</i> (R420).

## Obtaining IP Addresses and Domain Names

IP addresses and domain names are each administered by separate organizations.

To obtain an IP network number or use the services available to network members, contact an Internet Service Provider (ISP) or one of the following Regional Internet Registries (RIRs):

- The American Registry for Internet Numbers (ARIN) serves Canada, many Caribbean and North Atlantic islands, and the United States. You can contact ARIN at <http://www.arin.net>.
- Reseaux IP Europeens Network Communication Center (RIPE NCC) serves Europe, the Middle East, and parts of Africa. You can contact RIPE NCC at <http://www.ripe.net>.

- Asia Pacific Network Information Center (APNIC) serves Asia and the Pacific. You can contact APNIC at <http://www.apnic.net>.
- African Network Information Center (AFRINIC) serves Africa. You can contact AFRINIC at <http://www.afrinic.net>.
- Latin American and Caribbean Network Information Center (LACNIC) serves Latin American and Caribbean regions. You can contact LACNIC at <http://www.lacnic.net>.

#### NOTE

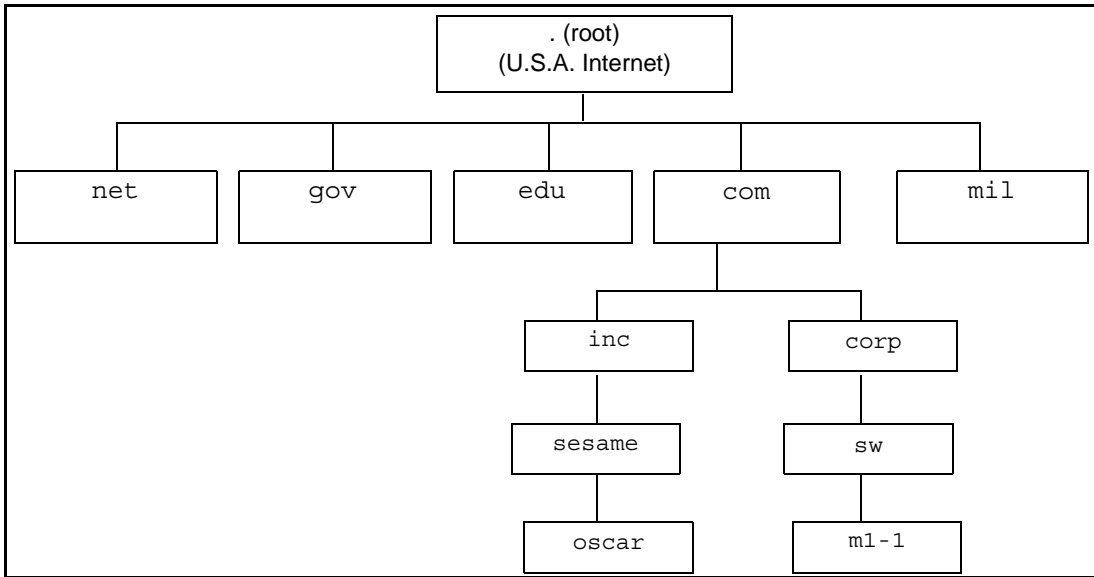
End users generally receive IP address space from their ISPs, not directly from a registry. Provider-independent (portable) addresses obtained directly from a registry generally cannot be routed globally.

To register for a domain name, contact an ISP or one of the many domain name registry organizations listed on the The Internet Corporation for Assigned Names and Numbers (ICANN) web page: <http://www.icann.org/registrars/accredited-list.html>. For more information about IP address assignments, contact Internet Assigned Numbers Authority (IANA), at <http://www.iana.org>.

## Domain Name Service

The Internet Domain Name Service (DNS) maps a symbolic host name to its numeric IP address. The domain name system is designed as a tree hierarchy. The first level or zone of the tree is the most inclusive and is called the *root*. The root is represented by a period (.). Every other zone in the tree extends from the root, is represented symbolically, and is separated from the preceding level by a period.

Figure A-9 shows a sample domain name hierarchy in which the root includes all members of the Internet in the United States. Since all commercial members of the Internet are members of the zone labeled `com`, `com` is a large subset of the root shown in the figure. The full symbolic name `m1-1.sw.corp.com` represents the local host name in the domain name hierarchy.



**Figure A-9. Sample Domain Name Hierarchy**

Each zone in the DNS is managed locally. Therefore, if you do not know the address of a remote host, you can query the root DNS server for the address of the DNS server in the zone in which you should start the search. For example, if your machine is the host named `m1-1.sw.corp.com` in [Figure A-9](#), and you do not know the IP address of the host named `oscar.sesame.inc.com`, you can query the DNS server for zone `com` for the IP address of the DNS server for zone `inc`. You can then query the DNS server for zone `inc` for the address of the DNS server for `sesame`, and so on, until you obtain the unknown address.

**NOTE** \_\_\_\_\_

The sample host names `m1-1.sw.corp.com` and `oscar.sesame.inc.com` are represented without the trailing period to represent the root. The trailing period is omitted by convention.

---

## Appendix B

# Sample STCP Configuration

This appendix shows a sample STCP configuration and includes all of the information required to build this type of configuration on sample module #m1.

This configuration has the following characteristics.

- It has multiple network interfaces configured for STCP, which automatically enables IP forwarding and allows the module to act as a router.
- It uses Class A network numbers.
- It assumes that #m1 is running STCP only and establishes the STCP command library path as well as the object and include library paths.
- It is an example IPv4 configuration.

This appendix contains the following sections.

- [“Sample Device Entries” on page B-3](#)
- [“Sample Database Files” on page B-7](#)

[Figure B-1](#) shows the sample STCP configuration.

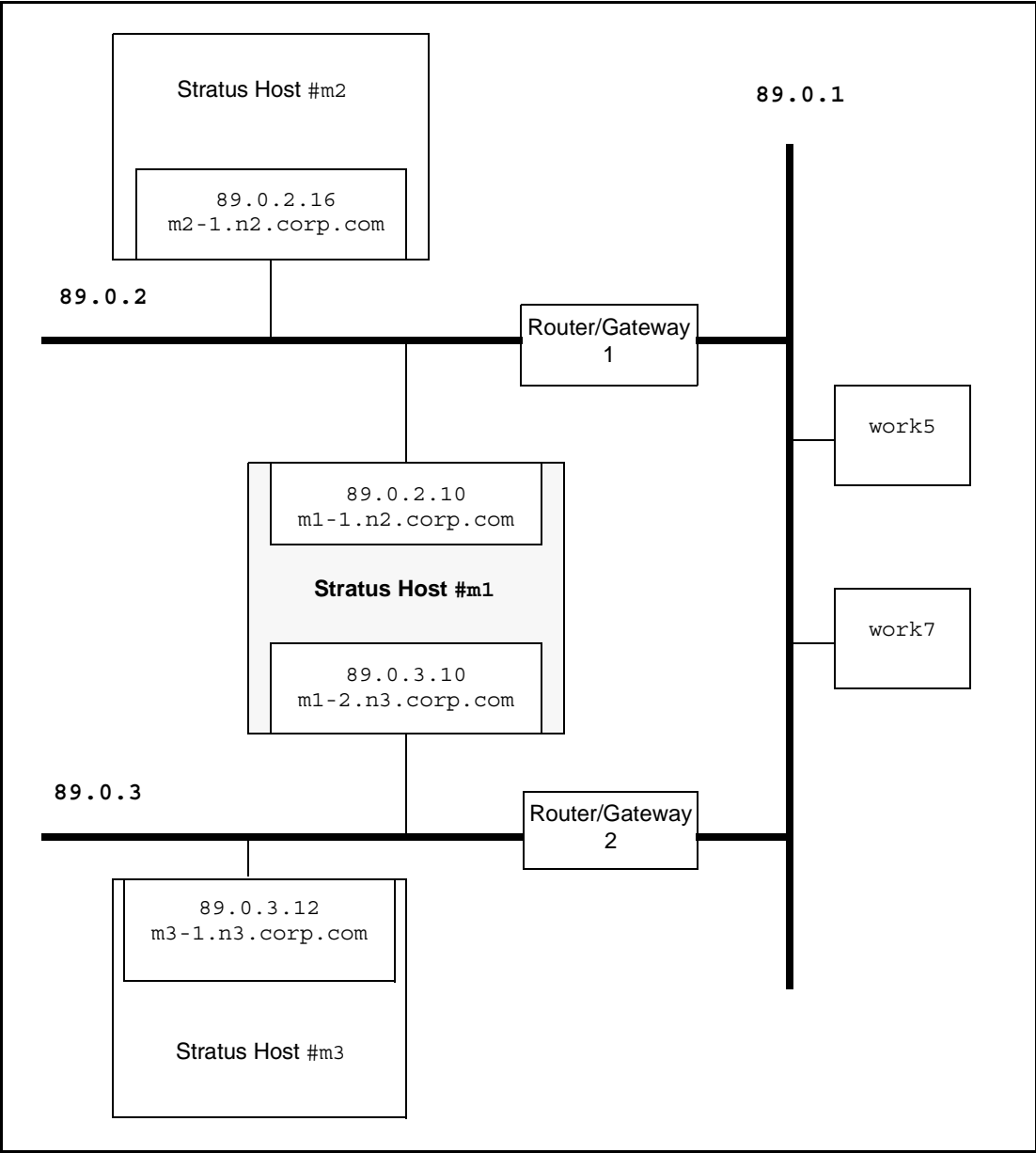


Figure B-1. Sample STCP Configuration (with Berkeley Numbers)



## Sample Device Entries

The sample device entries in this section configure an ftServer V 2404, V 4408, V 6408, or V 6512 system as module #m1 to run STCP and to accept TELNET connection requests using the STCP `telnetd` daemon process. [Figure B-2](#) shows a sample excerpt from the `devices.tin` file that includes entries for the following devices:

- network interfaces: two partnered embedded 10/100/1000 Ethernet PCI adapters and one simplex port on an Ethernet PCI adapter that you can insert.
- SDLMUX-group interfaces for the network interfaces
- STCP devices: protocol drivers (STCP, UDP, and IP), TELNET pipe devices, and TELNET incoming slave devices

### NOTE

**Do not enter the plus (+) sign that appears in the examples; it is a line-continuation character.** One line in a device entry can extend beyond one terminal-screen line. As you enter characters in a line that extends beyond one terminal-screen line, a plus sign automatically begins the second terminal-screen line to indicate that the line continues on a second terminal-screen line.

```
/* Entry for STCP protocol drivers */

/   =name                stcp.m1
    =module_name         m1
    =device_type         streams
    =access_list_name    stcp_access
    =streams_driver      stcp
    =clone_limit         5120

/   =name                udp.m1
    =module_name         m1
    =device_type         streams
    =access_list_name    stcp_access
    =streams_driver      udp
    =clone_limit         5120
```

*(Continued on next page)*

```

/      =name                ip.m1
      =module_name          m1
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       ip
      =clone_limit          1024

/      =name                loop.m1
      =module_name          m11
      =device_type          streams
      =access_list_name     stcp_access
      =streams_driver       loop
      =clone_limit          1024

/* Entries for network interfaces: */
/* two partnered embedded 10/100/1000 Ethernet PCI adapters and */
/* one simplex Ethernet PCI adapter */

/      =name                enetA.m1.10-6.0
      =module_name          m1
      =device_type          streams_pci
      =access_list_name     stcp_access
      =streams_driver       igb
      =hardware_address     '10/6/0'
      =parameters          '-partner #enetA.m1.11-6.0
+ -sdlnmux #sdlnmuxA.m1.10-6.0.11-6.0'
      =clone_limit          32

/      =name                enetA.m1.11-6.0
      =module_name          m1
      =device_type          streams_pci
      =access_list_name     stcp_access
      =streams_driver       igb
      =hardware_address     '11/6/0'
      =parameters          '-partner #enetA.m1.10-6.0
+ -sdlnmux #sdlnmuxA.m1.10-6.0.11-6.0'
      =clone_limit          32

```

*(Continued on next page)*

```
/* Sample entry for one simplex Ethernet PCI adapter */

/   =name                enetB.m1.11-3.0
    =module_name         m1
    =device_type         streams_pci
    =access_list_name    stcp_access
    =streams_driver      igb
    =hardware_address    '11/3/0'
    =parameters          '-sdlmux #sdlmuxB.m1.11-3.0'
    =clone_limit         32

/* Sample entries for the SDLMUX groups */

/   =name                sdlmuxA.m1.10-6.0.11-6.0 /* group on A*/
    =module_name         m1
    =device_type         streams
    =access_list_name    stcp_access
    =streams_driver      sdlmux
    =clone_limit         32

/   =name                sdlmuxB.m1.11-4.0 /* group on LAN B*/
    =module_name         m1
    =device_type         streams
    =access_list_name    stcp_access
    =streams_driver      sdlmux
    =clone_limit         32

/* Entry for TLI access-layer device driver and */
/* TELNET pipe devices */

/   =name                tli_al.m1
    =module_name         m1
    =device_type         streams
    =access_list_name    stcp_access
    =streams_driver      tli_term
    =minor_number        0
```

*(Continued on next page)*

```

    =name                tpipe_telnetd.m1
    =module_name          m1
    =device_type          streams
    =access_list_name     stcp_access
    =streams_driver       tpipe
    =minor_number         0

/
    =name                tpipe_admin.m1
    =module_name          m1
    =device_type          streams
    =access_list_name     stcp_access
    =streams_driver       tpipe
    =minor_number         1

/* Entries for two incoming slave devices (clonable) */

/
    =name                tli_inslv1m1
    =module_name          m1
    =terminal_type        ascii
    =device_type          window_term
    =login_slave          0
    =priv_terminal        1
    =dialup               0
    =clone_limit          1000

/
    =name                tli_inslv2m1
    =module_name          m1
    =terminal_type        ascii
    =device_type          window_term
    =login_slave          0
    =priv_terminal        1
    =dialup               0
    =clone_limit          1000
```

*(Continued on next page)*

```

/* Entries for one outgoing slave device (clonable) */

/   =name                tli_outslv1m1
    =module_name         m1
    =terminal_type       ascii
    =device_type         window_term
    =login_slave         0
    =priv_terminal       1
    =dialup              0
    =parameters          '-ip 172.16.99.49,969'
    =clone_limit         1000

```

**Figure B-2. Sample Excerpt from a `devices.tin` File**

## Sample Database Files

Figures B-3 through B-11 provide sample database files for the sample STCP configuration. Note that the sample files do not include the Stratus copyright information that appears in the template files.

Figure B-3 shows a sample `bootptab` database file.

```

# bootptab
# Example bootp table file, database for bootp daemon.
#
# Blank lines and lines beginning with '#' are ignored.
#
#
# home directory
#
>system>bootfiles
# default bootfile
default

%%

# The remainder of this file contains one line per client
# interface with the information shown by the table headings
# below. First search is on internet address if specified in
# incoming request. If not specified, hardware address is used.
# The 'host' name is tried as a suffix for the 'bootfile'
# when searching the home directory (e.g., bootfile.host).

```

*(Continued on next page)*

#	host	htype	haddr	iaddr	bootfile
	ncd	1	00-00-A7-11-34-64	89.0.10.66	bootfile.ncd
	m11	1	00-00-A8-8A-86-CB	89.0.10.64	bootfile.m11
	router1	1	08-00-39-00-60-0E	89.0.5.9	router.1_0
	router2	1	08-00-39-00-60-0F	89.0.5.10	router.1_0

**Figure B-3. Sample bootptab Database File**

Figure B-4 shows the `hosts` database file for the sample configuration.

127.0.0.1	localhost	loopback	# required
89.0.2.10	m1.n2.corp.com	m1	
89.0.3.10	m1-2.n3.corp.com	m1-2	
89.0.2.16	m2-1.n2.corp.com	m2-1	
89.0.3.12	m3-1.n3.corp.com	m3-1	
#			
89.0.1.5	work5.n1.corp.com	work5	# workstation
89.0.1.7	work7.n1.corp.com	work7	# workstation

**Figure B-4. Sample hosts Database File**

Figure B-5 shows the `inetd.conf` database file for the sample configuration.

```
# The parameters are in this order:
# service-name
# endpoint-type protocol
# wait-status
# uid
# server-program (full pathname)
# server-arguments
#
echo    stream tcp nowait root    internal  internal
echo    dgram  udp  wait  root    internal  internal
discard stream tcp nowait root    internal  internal
discard dgram  udp  wait  root    internal  internal
daytime stream tcp nowait root    internal  internal
daytime dgram  udp  wait  root    internal  internal
chargen stream tcp nowait root    internal  internal
chargen dgram  udp  wait  root    internal  internal
time    stream tcp nowait root    internal  internal
time    dgram  udp  wait  root    internal  internal
#
```

*(Continued on next page)*

```
# Uncomment the following lines only if you want to use the bootp
# or the tftpd server.
# You should be very careful since there are security issues with
# using tftp.
tftpd  dgram  udp  wait  root  >system>stcp>command_library>tftpd.pm
bootpd dgram  udp  wait  root  >system>stcp>command_library>bootpd.pm
#
```

**Figure B-5. Sample `inetd.conf` Database File**

Figure B-6 shows the `networks` database file for the sample configuration.

```
#
Corp_Net1  89.0.1 Net1
Corp_Net2  89.0.2 Net2
Corp_Net3  89.0.3 Net3
#
```

**Figure B-6. Sample `networks` Database File**

Figure B-7 shows the `ospfdconf` database file entries for the sample configuration.

```
#
# ABR Router
RTRID: 15.17.10.21  0

# Area 0.0.0.0 (backbone)
AREA: 0.0.0.0  0  0
AUTH: 0

# Area 0.0.0.2
AREA: 0.0.0.2  0  0
RANGE: 10.111.57.0 255.255.255.0
AUTH: 0

# Broadcast Interface
IF: 0.0.0.0 15.17.10.21  1  1  20  2  2  10  40  (password)

# Broadcast Interface
IF: 0.0.0.2 10.111.57.44  1  1  20  2  2  10  40  (password)
```

*(Continued on next page)*

```
# Virtual Link
VIRTUAL: 10.112.0.1 0.0.0.2 2 20 30 120 (password)

# Host Route
HOST: 0.0.0.2 10.111.103.55 10
```

**Figure B-7. Sample ospfdconf Database File**

Figure B-8 shows the protocols database file for the sample configuration.

```
# protocol number alias(es) # comment(s)
#
ip          0      IP      # internet pseudo protocol number
icmp        1      ICMP    # internet control message protocol
tcp         6      TCP     # transmission control protocol
udp         17     UDP     # user datagram protocol
```

**Figure B-8. Sample protocols Database File**

Figure B-9 shows the services database file for the sample configuration.

```
#
# Sample Network services, Internet style
#
#Name          port      Service
#
ftpdata        20/tcp
ftp            21/tcp      ftpd
telnet         23/tcp      telnetd
smtp           25/tcp
bootps         67/udp      bootpd
bootpc         68/udp      bootp
tftp           69/udp      tftpd
snmp           161/udp     snmpd
route          520/udp     router routed
echo           7/tcp
echo           7/udp
discard        9/tcp      sink null
discard        9/udp      sink null
daytime        13/tcp
daytime        13/udp
```

*(Continued on next page)*



```

chargen      19/tcp      ttttst source
chargen      19/udp      ttttst source
time         37/tcp      timserver
time         37/udp      timserver
finger       79/tcp
ntalk        518/udp
exec         512/tcp
domain       53/tcp      nameserver # name-domain server
domain       53/udp      nameserver
nb_nmsrv     137/udp      netbios-ns # netbios nameserver
nb_sssrv     139/tcp      netbios-ssn # netbios session server
nb_dgsrv     138/udp      # netbios datagram server
swat         901/tcp      # samba web configurator
sync_cfgd    29000/udp    sync_cfgd
swat         901/tcp      samba web configurator
rsn_incoming 85/tcp      samba web configurator

# user-defined services (for example, for incoming slaves)

m1slave1 950/tcp
m1slave2 951/tcp

```

**Figure B-9. Sample `services` Database File**

[Figure B-10](#) shows the `snmpconf` database file for the sample configuration.

```

sysContact = Jane_Doe (jdoe@mycorp.com)
sysName = M1 (m1.mycorp.admin.com)
sysLocation = GenericCorp, Everytown, MA, Eng, M1-2N
community_public = garlic
community_private = basil
community_proxy = parsley
community_regional = chives
community_core = cloves

```

**Figure B-10. Sample `snmpconf` Database File**

Figure B-11 shows the `telnet` service database file for the sample configuration.

```
telnet      window_term ""      "Telnet default login svce"  1 1
+tli_logm1
m1slave1   window_term ""      "m1 slave service 1"        0 0
+tli_inslv1m1
m1slave2   window_term "linger=10" "m1 slave 2 linger"  0 0
+tli_inslv2m1
rsn_incoming window_term "keepalive" "RSN Incoming Service" 0 1
+in_rsn_m2
```

**Figure B-11. Sample `telnet` service Database File**

---

## Appendix C

# Attaching an Application to a TELNET Slave Device

This appendix describes how to attach an OpenVOS host application to an STCP TELNET slave device in order to establish a slave connection with a remote TELNET client. The procedures for enabling these STCP TELNET slave connections depend on whether the remote client or the OpenVOS host initiates the TELNET connection.

This appendix contains the following sections.

- [“Enabling Incoming \(Client-Initiated\) Connections” on page C-1](#)
- [“Enabling Outgoing \(Host-Initiated\) Connections” on page C-5](#)

Slave connections that are initiated by a remote client are **incoming** (client-initiated) slave connections. Slave connections that are initiated by the OpenVOS host are **outgoing** (host-initiated) slave connections.

### NOTE

This appendix shows IPv4 protocol support.

## Enabling Incoming (Client-Initiated) Connections

Enabling an incoming (client-initiated) slave connection to an OpenVOS host application (service) on the local host involves the following steps.

1. Enable the STCP TELNET daemon process ([telnetd](#)) to listen at the TCP network port associated with the application and enable it to use an OpenVOS virtual device (an incoming slave device) that represents the application (defined as a TELNET incoming slave service).
2. Enable the OpenVOS host application to attach to and open the incoming slave devices configured to support it.
3. Establish a TELNET session between a remote TELNET client and the OpenVOS host running STCP by connecting to the network port at which the [telnetd](#) daemon process is listening for requests for the incoming slave service. (The

remote client must initiate a TELNET session that connects to the appropriate network port on the OpenVOS host.)

The following sections describe these steps in detail.

- “[Configuring `telnetd` for an Incoming Slave Connection](#)” on page C-2
- “[Opening a TELNET Incoming Slave Device](#)” on page C-4
- “[Establishing a TELNET Session on the Client](#)” on page C-5

#### NOTE

An incoming slave connection request requires one or more STCP TELNET incoming slave device entries in the `devices.tin` file. For information about how to define an STCP TELNET incoming slave device entry in the `devices.tin` file, see “[Defining STCP TELNET Incoming Slave Devices](#)” on page 4-20.

## Configuring `telnetd` for an Incoming Slave Connection

In order for `telnetd` on the OpenVOS host to establish a TELNET session for an incoming slave connection, `telnetd` must be configured to listen at the network port specified for the incoming slave service. In addition, to start the TELNET session, `telnetd` must be able to find the STCP TELNET incoming slave device that has been attached to and opened by an OpenVOS host application (using OpenVOS subroutines such as `s$attach_port` and `s$open`). Before you initially configure STCP and start the `telnetd` daemon process (typically started from the `start_stcp.cm` command macro instead of from command level), the following requirements must be met.

- In the `services` database file, specify an Internet port number at which `telnetd` listens for incoming slave connection requests for the incoming slave service. You should specify a network port that other services are not currently using.
- In the `telnetSERVICE` and `services` database files, specify the service name of the service.
- In the entry for the service in the `telnetSERVICE` database file, specify the clonable device name or device prefix for the service in the `device_prefix` field (the last field in the entry). If you specify a clonable device name, it must match the entire device name specified in the `name` field of the device entry for the clonable incoming slave device; if you specify a device prefix, it must match the device prefix of the name specified in the `name` field of multiple incoming slave device entries for the service.

- In the entry for the service in the `telnet` service database file, the value 0 must appear in the `login_slave` field to specify that the service is a slave service. The device entry for each device that uses the service must also specify the value 0 in the `login_slave` field.
- In the entry for the service in the `telnet` service database file, the value you specify in the `privileged` field must match the value specified in the device entry or entries associated with the service (the value in the `priv_terminal` field).

Note that the preceding rules also apply if you are adding an incoming slave service using the `telnet_admin` command. (See [Chapter 9](#) for a description of the `telnet_admin` command.)

The following example is a sample excerpt from a `telnet` service database file. It contains three entries for incoming slave services (services `m1slave1`, `m1slave2`, and `m1slave3`). These services use clonable device names (`tli_inslv1m1`, `tli_inslv2m1`, and `tli_inslv3m1`). Two employ the TELNET linger option.

```
m1slave1 window_term "keepalive nodelay" "slave1 service" 0 0 tli_inslv1m1
m1slave2 window_term "keepalive linger=10 nodelay" "slave2 linger" 0 0 tli_inslv2m1
m1slave3 window_term "keepalive linger=15 nodelay" "slave3 linger" 0 0 tli_inslv3m1
```

The following example is a sample excerpt from a `services` database file. These services correspond to the preceding `telnet` service file entries. Note that the first field lists a service as it is named in the first field of the `telnet` service database file.

```
#
m1slave1      950/tcp
m1slave2      951/tcp
m1slave3      952/tcp
```

The `m1slave1` entry in both the `telnet` service and `services` database files enables the `telnetd` daemon process to listen at port 950 on the host for incoming slave connection requests. The host application must already be executing and must have attached to and opened a device (in this case, using its clonable device name `#tli_inslv1m1`).

When the remote client sends a connection request to port 950 on the host and the `telnetd` daemon process receives the connection request, `telnetd` checks the `telnet` service and `services` database files to associate port 950 with service `m1slave1`, and establishes a TELNET session by attaching the request to an open slave device whose device entry has the clonable device name `tli_inslv1m1` (or device prefix).

You can add a TELNET incoming slave service while the `telnetd` daemon process is running by issuing the `telnet_admin` command, as shown in the following example. By default, the service is privileged with the command; if you do not want the service to be privileged, specify `-no_privileged` on the command line.

```
telnet_admin add -service m1slave1 -device_prefix  
+ tli_inslv1m1 -description 'm1slave1 plain'  
+ -no_login -services_port 950
```

#### NOTE

The symbol `+` (the line continuation character) in the sample entry indicates that the entry required more than a single line on the terminal screen.

The `telnet_admin` command automatically updates the `telnet-service` and `services` files. For more information, see the description of the `telnet_admin` command in [Chapter 9](#). For more information about the `telnet-service` and `services` database files, see [Chapter 5](#).

## Opening a TELNET Incoming Slave Device

To enable the OpenVOS host application to open the appropriate STCP TELNET slave devices, you must include in the application an OpenVOS subroutine call that opens a slave device using the clonable device name specified in the `name` field of a device entry (which matches the name specified for the `device_prefix` argument of the `telnet-service` database file). For example, if the clonable device name in the device entry is `tli_inslv1m1`, the application could open multiple cloned devices by using the name `#tli_inslv1m1` in each call. The host application must open the slave device before the remote client's connection can be accepted.

#### NOTE

While the device entry does not precede the device name with the `#` sign, an application must use the `#` sign when specifying the device name.

## Establishing a TELNET Session on the Client

To establish a TELNET session between a remote client (for example, a client on a workstation) and the host, you issue the appropriate `telnet` command from the workstation, (or computer or terminal server). You must specify the IP address (or host name) of the host to which the client application will connect. You must also specify the network port number at which the appropriate STCP `telnetd` daemon process on the host is listening for incoming slave connection requests. For example, if you want to use the TELNET utility on the client to connect to port 950 on the STCP host `mktg` whose IP address is `172.16.2.56`, issue the following command.

```
telnet 172.16.2.56 950
```

If the remote client is an STCP TELNET client running on a Stratus module, see the *OpenVOS STREAMS TCP/IP User's Guide* (R421) for more information about issuing the STCP `telnet` command. If the remote TELNET client is not running on a Stratus module, see the appropriate documentation for the workstation, computer, or terminal server.

## Enabling Outgoing (Host-Initiated) Connections

Enabling an outgoing (host-initiated) slave connection from an OpenVOS host application attached to a slave device on the local host to a remote client (service) involves the following steps.

1. Enable the `telnetd` daemon process to find an outgoing slave device on the local host that represents the remote service (that is, configure an outgoing slave device that defines a clonable device name or common prefix in its device entry to accommodate the remote service).
2. Execute a server application on the remote host at the IP address and network port that match the IP address and network port specified in the `parameters` field of an STCP TELNET outgoing slave device entry. (If the IP address is for a network printer, note that the `parameters` field must include the argument `-tcp_only`.)
3. Enable the host application to attach to and open an STCP TELNET outgoing slave device defined for the remote host.

The following sections describe these steps in detail.

- [“Configuring Outgoing Slave Connections” on page C-6](#)
- [“Executing a Server Application on the Remote Host” on page C-7](#)
- [“Opening a TELNET Outgoing Slave Device” on page C-8](#)

## Configuring Outgoing Slave Connections

In order for the `telnetd` daemon process on the local OpenVOS host to establish a TELNET session for outgoing slave connections, `telnetd` must be able to find an outgoing slave device that has been attached to and opened by a local OpenVOS host application (using OpenVOS subroutines such as `s$attach_port` and `s$open`) and that is reserved to accommodate a server application on the remote host. Before you initially configure STCP and start the `telnetd` daemon process (typically started from the `start_stcp.cm` command macro instead of from command level), the following requirements must be met.

- Use the `name` field of a device entry to specify the clonable device name or device prefix for an outgoing slave device associated with the server application on the remote host. This enables the server process to establish the TELNET connection by selecting only those devices that have been attached to and opened by a local OpenVOS application and that have a matching device name or device prefix.
- Use the `-ip` argument in the `parameters` field of the local outgoing slave device entry (entries) to specify both the IP address and Internet port that the `telnetd` daemon process and TLI software use to make the outgoing connection.
- Use the `login_slave` field of the local outgoing slave device entry to specify the value `0`, which indicates that the device is a slave. You must specify the value `0` in order to enable the `telnetd` daemon process and the TLI software to initiate the connection.

### NOTE

You do not place STCP TELNET outgoing slave services in either the `telnet-service` or `services` file. When an OpenVOS host application attaches to and opens an outgoing slave device, the `telnetd` daemon process checks the device entry for the specified device and then checks the service information (for example, the clonable device name or prefix) in the `telnet-service` file. If `telnetd` does not find the corresponding information in the file, it knows that the service is available remotely and initiates the connection to the server application on the remote host based on the IP address and network port (also listed in the device entry).



The following sample device entries define two outgoing slave devices on sample module #m1 that use a common device prefix (and unique suffixes). In the entries, the `telnetd` daemon process on #m1 can use one of these outgoing slave devices to connect to a network printer (a remote service), represented **remotely** by the service name `m2printer1` and the IP address `172.16.3.25` at port `958`. In order for the `telnetd` daemon process to establish the connection, the server application on the remote host **must** be listening at the matching IP address for the connection request from the host.

```
/  =name          tli_outslv1m1.1
   =module_name   m1
   =terminal_type ascii
   =device_type   window_term
   =login_slave   0
   =parameters    '-tcp_only -ip 172.16.3.25,958'

/  =name          tli_outslv1m1.2
   =module_name   m1
   =terminal_type ascii
   =device_type   window_term
   =login_slave   0
   =parameters    '-tcp_only -ip 172.16.3.25,958'
```

For information about the fields in this type of device entry, see [“Defining STCP TELNET Outgoing Slave Devices” on page 4-26](#). For more information about the `telnetd` daemon process, see the description of `telnetd` in [Chapter 8](#).

## Executing a Server Application on the Remote Host

A server application **must** be executing on the remote host at the specified IP address and listening on the specified network port in order for the `telnetd` daemon process on the local host to establish a TELNET session with the remote host. (The IP address of the remote host and the network port at which the server application is executing are specified by the `-ip` argument of the parameters field in the `devices.tin` file entry. The `telnetd` daemon process and the TLI software automatically initiate a connection to the remote host and server application whose IP address and network port are specified by the `-ip` argument of the parameters field in the device entry.)

If the remote host is a Stratus module, you must use the appropriate STCP or OS TCP/IP functions to enable the server application to execute at the specified network port. For information about the STCP functions, see the *OpenVOS STREAMS TCP/IP Programmer's Guide* (R420).

## Opening a TELNET Outgoing Slave Device

To enable the OpenVOS host application to open the appropriate STCP TELNET slave devices, you must include in the application an OpenVOS subroutine call (using OpenVOS subroutines such as `s$attach_port` and `s$open`) that opens a slave device by specifying the clonable device name specified in the outgoing slave device entry, and an IP address and network port specified in the `parameters` field of the device entry.

If you are using a clonable outgoing slave device entry to accommodate the server application on the remote host, the host application could attach to and open the clonable device multiple times using the clonable device name (for example, `#tli_outslv1m1`) in each call. The `telnetd` daemon process could then use these cloned devices, as long as the device entry also specifies the IP address and network port for the server application on the remote host.

---

## Appendix D

### OSPF Debug Errors

[Table D-1](#) describes the messages returned by the `omon` subcommand, `list_errors`. Not all messages are error messages; some messages provide information about the behavior of the module or router running the `ospfd` daemon process and of the network during the current session of OSPF.

**Table D-1. Errors Displayed by the `list_errors` Subcommand** *(Page 1 of 4)*

Message	OSPF Packet Type	Description
IP: Bad OSPF pkt type	Any	The packet type is not a correctly specified type. OSPF packet types are Hello, Database Description, Link State Request, Link State Update, and Link State ACK.
IP: Bad IP Dest	Any	The destination address associated with the packet is not that of a known interface and is not an IP multicast address.
IP: Bad IP proto id	Any	The IP header of the packet contained an incorrect protocol number. The OSPF protocol number is 89.
IP: Pkt src = my IP addr	Any	The source IP address in the packet is the same as that of the system being monitored.
OSPF: Bad OSPF version	Any	The OSPF version specified in the packet is not the same as that of the host system. OpenVOS modules use OSPF version 2.
OSPF: Bad OSPF checksum	Any	The checksum specified in the packet does not match the calculated value.
OSPF: Bad intf area id	Any	The area ID calculated from the source IP address of the packet does not match that of the local area.
OSPF: Area mismatch	Any	Only a border router can handle the packet. If the system being monitored is not a border router, this packet will be discarded.

**Table D-1. Errors Displayed by the list\_errors Subcommand** (Page 2 of 4)

Message	OSPF Packet Type	Description
OSPF: Bad virt link info	Any	The virtual link associated with the packet is not listed in the OSPF configuration on the monitored router.
OSPF: Auth type != area type	Any	The authentication type specified in the packet (for example, password) does not match the authentication type in the OSPF configuration on the monitored router.
OSPF: Auth key != area key	Any	The authentication key specified in the packet does not match the authentication key in the OSPF configuration on the monitored router.
OSPF: Packet is too small	Any	The length of the OSPF header for the received packet is less than 24 bytes.
OSPF: Packet size > IP length	Any	The length of the OSPF header is greater than the length of the IP packet and therefore invalid.
OSPF: Transmit bad	Any	The transmission of the OSPF packet failed.
OSPF: Received on down IF	Any	The interface on which the packet was received is in a DOWN state.
Hello: IF mask mismatch	Hello	The subnet mask specified in the packet does not match the subnet mask in the OSPF configuration on the monitored router.
Hello: IF hello timer mismatch	Hello	The hello timer interval specified in the packet does not match the hello timer interval in the OSPF configuration on the monitored router.
Hello: IF dead timer mismatch	Hello	The dead timer interval specified in the packet could does not match the dead timer interval in the OSPF configuration on the monitored router.
Hello: Extern option mismatch	Hello	The external option in the packet did not match that of the area receiving the packet.
Hello: Nbr Id/IP addr confusion	Hello	The packet specifies that the neighbor has the same router ID as the monitored module has.
Hello: Unknown Virt nbr	Hello	The interface for which the packet is destined has been identified to be a virtual type. However the area ID for the transit area specified in the packet does not match the area ID for the transit area in the OSPF configuration on the monitored router.

**Table D-1. Errors Displayed by the list\_errors Subcommand (Page 3 of 4)**

Message	OSPF Packet Type	Description
Hello: Unknown NBMA nbr	Hello	The neighbor specified in the packet is not listed as a NBMA type in the OSPF configuration on the monitored router.
DD: Unknown nbr	Database Description	The IP address specified in the packet identifies an unknown neighbor.
DD: Nbr state low	Database Description	The neighbor specified in the packet is in a DOWN state.
DD: Nbr's rtr = my rtrid	Database Description	The packet specifies that the neighbor has the same router ID as the monitored module has.
DD: Extern option mismatch	Database Description	The external option specified in the packet did not match that of the area receiving the packet
Ack: Unknown nbr	Link State Acknowledgement	The neighbor's IP address in the packet identifies an unknown neighbor.
Ack: Nbr state low	Link State Acknowledgement	The neighbor specified in the packet is in a NOEXCHANGE state.
LS Req: Nbr state low	Link State Request	The neighbor specified in the packet is in a NOEXCHANGE state.
LS Req: Unknown nbr	Link State Request	The IP address in the packet identifies an unknown neighbor.
LS Req: Empty request	Link State Request	The packet has been determined to be empty and would be discarded.
LS Req: Bad pkt	Link State Request	The packet has been identified as unusual, for one of many reasons, and would be discarded.
LS Update: Nbr state low	Link State Update	The neighbor specified in the packet is in a NOEXCHANGE state.
LS Update: Unknown nbr	Link State Update	The IP address in the packet identifies an unknown neighbor.
LS Update: Newer self-gen LSA	Link State Update	Not used.
LS Update: Bad LS chksum	Link State Update	The calculated link state update checksum does not match the value specified in the packet.

**Table D-1. Errors Displayed by the list\_errors Subcommand** (Page 4 of 4)

Message	OSPF Packet Type	Description
Ls Update: less recent rx	Link State Update	The packet has been identified as unusual, for one of many reasons, including an out-of-sequence packet.
Ls Update: Unknown type	Link State Update	The packet type is not one of the following link state update types: Router LSA, Network LSA, Network Summary LSA, ASBR Summary LSA, AS External LSA.

---

## Appendix E

# OSPF Error Messages

Error conditions occur while the OSPF daemon is starting up and during normal operation of the OSPF daemon. A misconfigured `ospfdconf` file is a common cause of errors. However, some errors are caused by the failure of the protocol stack, network problems, or problems with other routers in the autonomous system (AS).

If you configure the OSPF daemon to log error messages, you can examine them in the OSPF error log file `(master_disk)>system>stcp>logs>ospfd.io`. You can also configure the OSPF daemon to display error messages on your terminal.

The following errors indicate a misconfiguration in the `ospfdconf` file. The occurrence of such an error causes the OSPF daemon to terminate. The errors are listed alphabetically by error message within this section.

`Can't open config file file_name`

Indicates that the OSPF daemon could not open the `ospfdconf` file specified by *file\_name*. Make sure that the file exists and that the OSPF daemon has permission to read from the file.

`conf: address: unknown mask`

Indicates an error in a `RANGE` entry. Make sure that the *subnet\_mask* fields specify values in dotted decimal notation.

`conf: address: unknown net`

Indicates an error in a `RANGE` entry. Make sure that the *IPaddr* fields specify values in dotted decimal notation.

`conf: area_id: area_id is bad`

Indicates an error in an `AREA` entry. Make sure that the *area\_ID* fields specify values in dotted decimal notation.

`conf: HOST: area id area_id not found`

Indicates an error in a `HOST` entry. Make sure that the `AREA` entry for the host route appears before the `HOST` entry.

Stratus does not support point-to-point interfaces on OpenVOS modules.

conf: HOST: *IP\_address* invalid host address

Indicates an error in a HOST entry. Make sure that the *IPaddr* field specifies a value in dotted decimal notation.

Stratus does not support point-to-point interfaces on OpenVOS modules.

conf: illegal input *text* in file: *file\_name*

Indicates that the *ospfdconf* file specified by *file\_name* contains an invalid statement, identified by *text*. Make sure that the first character on each line (except blank lines) is either a pound (#) sign, to denote a comment line, or a valid keyword.

conf: interface: *area\_ID* is bad

Indicates an error in an IF entry. Make sure that the *area\_id* field specifies a value in dotted decimal notation.

conf: interface: area id *Area\_ID* is not found

Indicates an error in an IF entry. Make sure that the AREA entry appears before the IF entry.

conf: interface: *IP\_address* unknown addr

Indicates an error in an IF entry. Make sure that the *IPaddr* field specifies a value in dotted decimal notation.

conf: interface: illegal type *type*

Indicates an error in an IF entry. Make sure that the interface type is 1, to specify broadcast.

Although 2, to specify NBMA, or 3, to specify point-to-point, are valid values and will not cause error messages, Stratus does not support NBMA or point-to-point interfaces on OpenVOS modules.

conf: nbma: *IP\_address* unknown host

Indicates an error in an NBMA entry. Make sure that the *IPaddr* field specifies a value in dotted decimal notation.

Stratus does not support NBMA interfaces on OpenVOS modules.

conf: nbma: if is not defined to be nonbroadcast

Indicates an error in an IF or NBMA entry. Make sure that the *type* field in the IF entry for the interface is configured as 2, to specify NBMA.

Stratus does not support NBMA interfaces on OpenVOS modules.



`conf: nbma: interface id IP_address not found`

Indicates an error in an NBMA entry. Make sure that an IF entry for the interface appears before the NBMA entry.

Stratus does not support NBMA interfaces on OpenVOS modules.

`conf: nbma: IP_address: unknown if`

Indicates an error in an IF or NBMA entry. Make sure that the *IPaddr* fields in both entries specify values in dotted decimal notation.

Stratus does not support NBMA or point-to-point interfaces on OpenVOS modules.

`conf: rtr_id: router_id: unknown host`

Indicates an error in the RTRID entry. Make sure that the *router\_id* field specifies a value in dotted decimal notation.

`conf: transit area_id: area_id is bad`

Indicates an error in a VIRTUAL entry. Make sure that the *area\_id* fields in VIRTUAL entries are specified in dotted decimal notation.

`conf: virtual: nbr: transit area area_id not configured.`

Indicates that the *area\_id* field in a VIRTUAL entry specifies an area that has not been defined in the *ospfdconf* file. Make sure that the AREA entry for the transit area appears before the VIRTUAL entry.

`conf: virtual: nbr: weird nbr id`

Indicates an error in a VIRTUAL entry. Make sure that the *neighbor* fields in VIRTUAL entries are specified in dotted decimal notation.

`conf: virtual: VL defined but backbone hasn't`

Indicates that a virtual link has been configured but that the backbone area has not been configured. Make sure that an AREA entry for the backbone appears before any VIRTUAL entries.

`interface: attempt to define BkBone IF without defining BkBone`

Indicates that an interface belonging to the backbone is defined before the backbone. Make sure that an AREA entry for the backbone appears before any IF entries.

`interface: cost for tos 0 not set`

Indicates that the *metric* field specified in an IF entry specifies a value less than 1. Make sure that all *metric* fields specify a value equal to or greater than 1.

`interface: illegal priority`

Indicates that the *priority* field in an IF entry specifies an invalid value. Make sure that the *priority* fields specify values of less than 256.

Intfs are not on machine: (ifs count nintf ospfcount)

Indicates that more IF entries exist in the ospfdconf file than in the IP configuration. For *count*, the error message reports the number of interfaces in the system and for *ospfcount*, the error message reports the number of interfaces configured in the ospfdconf file. Make sure that the IP addresses configured for the module match those in the ospfdconf file.

>> looking for strings: ATTRTR: POLLINT: or #

Indicates an error in an NBMA definition. Make sure that the NBMA definition starts with an NBMA entry, is followed by one or more ATTRTR entries, and ends with a POLLINT entry.

Stratus does not support NBMA interfaces on OpenVOS modules.

looking for strings: RANGE: AUTH: or #

Indicates an error in an area definition. Make sure that the area definition starts with an AREA entry, is followed by one or more RANGE entries, and ends with an AUTH entry.

ospf: IF *IP\_address* is defined by machine to be broadcast or point-to-point

Indicates that the configuration of an interface in the ospfdconf file specifies a value of 2 in the *type* field to indicate NBMA, but the IFF\_BROADCAST or IFF\_POINTOPOINT flag is set in the kernel.

Stratus does not support NBMA interfaces on OpenVOS modules.

ospf: IF *IP\_address* not defined by machine to be point-to-point

Indicates that the configuration of an interface in the ospfdconf file specifies a value of 3 in the *type* field to indicate point-to-point, but the IFF\_POINTOPOINT flag is not set in the kernel.

Stratus does not support point-to-point interfaces on OpenVOS modules.

ospfsock: IF *IP\_address* not configured to be multicast

Indicates that the configuration of an interface in the ospfdconf file specifies a value of 1, to indicate broadcast, but the IFF\_MULTICAST flag is not set in the kernel.

The following errors are due to system failures or internal problems. The occurrence of such an error causes the OSPF daemon to terminate.

Can't open *file\_name*

Indicates that the OSPF daemon was unable to open or create a file to contain its process id. The file is usually

(master\_disk)>system>stcp>logs>ospfd.pid.

Fatal error: calloc returned null

Indicates that a request to the system to allocate memory failed.

fcntl (F\_SETFL, FNDELAY) fails

Indicates a failure to set nonblocking mode on a raw IP socket.

> multicast: socket (AF\_INET, SOCK\_RAW) fails

Indicates a failure to open a raw IP socket.

ospf\_rxpkt: select: error\_message

Indicates that the select call failed. *error\_message* is text that describes the cause of the failure, generally expiration of a time-out.

recvfrom socket\_id: error\_message

Indicates that the recvfrom call failed. *socket\_id* specifies the socket and *error\_message* is text that describes the cause of the failure, generally expiration of a time-out.

setsockopt IP\_MULTICAST\_IF failed name

Indicates a failure to set IP multicast on an interface specified by *name*.

setsockopt IP\_MULTICAST\_LOOP failed name

Indicates a failure to switch off looping back of multicast packets on an interface specified by *name*.

vs=S\_OPEN

((master\_disk)>system>kernel\_loadable\_library>ip.cp.pm, O\_RDWR,  
0): fails

Indicates that the details of an interface could not be obtained from the IP\_STREAMS driver.

The following errors do not cause the OSPFD daemon to terminate. The error messages indicate a misconfiguration or a problem with the IP protocol stack, the network, or other routers within the AS.

ospf\_txpkt sendto: ifspfndx index IP\_address: error

Indicates that an attempt to send an OSPF packet using the sendto function failed. Text in the error message includes *index*, which specifies the index number of the interface, *IP\_address* that specifies the IP address of the destination, and *error* that describes the error.

Rxlog: error

Indicates that a bad OSPF packet has been received.

*error* is text that describes the error. Do not confuse this with a normal receive packet log message, which contains one of the `Multicast` or `Unicast` keywords, in one of the following forms:

```
Rxlog: Multicast message  
or  
Rxlog: Unicast message
```

The following errors might occur while you are using `omon` to manage OSPF on a remote system.

Bad Connect

Indicates that the OSPF daemon could not set up a TCP connection for the `omon` command.

Bad mrequest type

Indicates an error in the packet.

## Related Information

For information about configuring OSPF to log error messages, see the following documentation.

- the description of the `ospfd` command in [Chapter 8](#)
- “[Problem 6: Module Cannot Route Packets](#)” on page 10-6
- the description of the `omon` command in [Chapter 9](#)

For information about the configuration of the `ospfdconf` file, see [Chapter 5](#).

---

## Appendix F

### The MST Server

STCP supports the Multisession TELNET (MST) server (`telnet_msd.pm`) in addition to the standard TELNET server (`telnetd.pm`). The MST server is based on the standards document RFC 854, which is presented in the *Defense Data Network (DDN) Protocol Handbook*, Volume 2, *DARPA Internet Protocols* (December 1985).

#### NOTICE

The standard STCP TELNET server process (`telnetd`) and the MST server process (`telnet_msd`) must not be listening for requests at the same network port. If you are running both of these server processes and you execute both the `telnet` command and the `telnet_msd` command, you must specify a different value for the port argument of each command (`-network_port` for the `telnet_msd` command and `port_number` for the `telnet` command).

#### NOTES

1. STCP supports the MST server for legacy applications only. Do **not** use the MST server with new applications.
2. References to TELNET in this appendix refer to the MST server (`telnet_msd.pm`). References to TELNET in all other chapters and appendixes in this manual refer to the standard TELNET server (`telnetd.pm`).
3. The `ftpd` daemon process supports IPv4 and IPv6 addresses.

This appendix, which contains the following sections, describes configuration requirements for the MST server.

- “[Configuring TELNET Virtual Terminal Devices](#)” on page F-2

- [“Starting the MST Server Process Automatically” on page F-6](#)
- [“The `telnet\_msd` Command” on page F-7](#)

## Configuring TELNET Virtual Terminal Devices

The MST server uses TELNET virtual terminal (vterm) devices. You must configure these devices in the `devices.tin` file, then re-create and broadcast the `devices.table` file. You must also start the vterm driver. The following sections describe these procedures.

- [“Defining TELNET Virtual Terminal Devices” on page F-2](#)
- [“Creating a `devices.table` File” on page F-5](#)
- [“Broadcasting the `devices.table` File” on page F-5](#)
- [“Issuing the `configure\_comm\_protocol` Command” on page F-6](#)
- [“Confirming Device Configuration” on page F-6](#)

## Defining TELNET Virtual Terminal Devices

The MST server process on your module uses a vterm device to connect each incoming TELNET connection request to a OpenVOS process. One vterm device must be defined in the `devices.tin` file for each concurrent TELNET session. Therefore, you should define the number of vterm devices required to accommodate the maximum number of TELNET sessions that can run concurrently on your module.

The MST server process shares with other OpenVOS processes the vterm devices that have been defined in the `devices.tin` file. To ensure that vterm devices are available for processes other than TELNET sessions, you can define subsets of vterm devices in the `devices.tin` file by using a naming convention (for example, a specific prefix) when specifying values for the `name` field. For example, you can name one subset of vterm devices `vterm.1.1`, `vterm.1.2`, and so on, and another subset of vterm devices `telnet.1.1`, `telnet.1.2`, and so on. You then specify the appropriate value (for example, `telnet*`) for the `-vterm_sturname` argument of the `telnet_msd` command. This ensures that the MST server process uses only the vterm devices designated for TELNET sessions. The default value for the `-vterm_sturname` argument is `telnet*`, which ensures that the server process uses vterm devices `telnet.1.1`, `telnet.1.2`, and so on.

When a process tries to attach to a vterm device, it searches the `devices.table` file (which is derived from the `devices.tin` file) sequentially from the top of the file for the first available vterm device. To ensure that the TELNET vterm devices are not attached by other processes until all other vterm devices have been used, you must position the definitions of the TELNET vterm devices **after** the definitions of the other vterm devices in the `devices.tin` file.

There are two types of vterm devices: login vterm devices and slave vterm devices. A *login vterm device* enables a remote client to log in to a Stratus module and connect to any available process (or application). A *slave vterm device* is controlled by a specific process (or application).

The following sections provide additional information about `devices.tin` file entries for TELNET vterm devices.

- [“Fields Used in Entries for TELNET Vterm Devices” on page F-3](#)
- [“Sample Entries for TELNET Vterm Devices” on page F-5](#)

### Fields Used in Entries for TELNET Vterm Devices

The following fields in the `devices.tin` file enable you to define a TELNET vterm device. All fields except the `parameters` field apply to both login and slave vterm devices. The `parameters` field applies to slave vterm devices only.

► `name`

Specify the name assigned to the virtual terminal. No two virtual terminals can have the same name.

#### NOTE

The name of a TELNET vterm device should begin with the prefix `telnet` because, by default, the MST server process searches for a vterm device with that prefix. The `devices.tin` file may define many vterm devices; therefore, you should group the TELNET vterm device definitions together and position them directly **after** the other vterm device definitions. By specifying the `telnet` prefix in the `name` field and positioning the TELNET vterm device definitions after the other vterm device definitions, you prevent other processes that use vterm devices from choosing one of your TELNET vterm devices. To enable the MST server process to search for vterm devices whose names begin with other prefixes, specify a value for the `-vterm_sturname` argument of the `telnet_msd` command that starts that server process.

► `module_name`

Specify the name of the module associated with the vterm device.

► `terminal_type`

Specify the type of remote terminal. You should specify the value `ascii` so that the software recognizes any type of remote ASCII terminal.

- ▶ `device_type`  
Specify the type of device being configured. For a vterm device, you must specify the value `vterm`.
- ▶ `baud`  
Specify the baud rate associated with the vterm device. You should specify the default value `9600`.
- ▶ `parity`  
Specify the parity supported by the remote host. You must specify the value `space`.
- ▶ `login_slave`  
Specify how the remote terminal associated with the vterm device will function. If the remote terminal connects to a vterm device that is configured as a login vterm device, the remote terminal can log in to the Stratus module. If it connects to a vterm device that is configured as a slave vterm device, the remote terminal is controlled by the process (or application).  
  
Specify the value `1` to configure a login vterm device; specify the value `0` to configure a slave vterm device.
- ▶ `priv_terminal`  
Specify whether or not a privileged process can log in to the remote terminal associated with a login vterm device. Specify the value `1` for yes (valid only for login vterm devices); specify the value `0` for no (valid only for slave vterm devices).
- ▶ `parameters`  
Specify a value, called the slave ID, for slave vterm device entries only. (This field is ignored in entries for login vterm devices.) The slave ID contains up to 12 characters, one or more of which are considered the slave ID prefix. In the sample slave vterm device entry, the `parameters` field contains the value `'id=service1'`; the prefix is `id=`. You must specify at least the slave ID prefix in this field; anything else is optional. Note that if you want to restrict an MST server process to devices with a particular `id=` value, you must specify that value with the `-vterm_slave_id` argument of the `telnet_msd` command. Note also that you must enclose in apostrophes the value that you specify for the `parameters` field.



## Sample Entries for TELNET Vterm Devices

The following sample `devices.tin` file entry shows the fields and values required to define a TELNET login vterm device.

```
/      =name                telnet.1.1
      =module_name          m3
      =terminal_type        ascii
      =device_type          vterm
      =baud                  9600
      =parity                space
      =login_slave          1
      =priv_terminal         1
```

The following sample `devices.tin` file entry shows the fields and values required to define a TELNET slave vterm device.

```
/      =name                telnet.1.2
      =module_name          m3
      =terminal_type        ascii
      =device_type          vterm
      =baud                  9600
      =parity                space
      =login_slave          0
      =priv_terminal         0
      =parameters            'id=service1'
```

## Creating a `devices.table` File

After you define the required STCP devices in the `devices.tin` file, you must convert the configuration information to a machine-executable `devices.table` file. You issue the `create_table` command to generate a new `devices.table` file.

From the directory `(master_disk)>system>configuration`, issue the following command.

```
create_table devices
```

## Broadcasting the `devices.table` File

After you generate the `devices.table` file, you must copy it to the `(master_disk)>system` directory on all modules in the system (including the current module). The following command copies (broadcasts) the updated `devices.table` file to all modules in the system.

```
broadcast_file devices.table >system
```

The new `devices.table` file does not take effect until the next bootload. However, you can issue the `configure_devices` command on each module in the system so that the operating system on the current module immediately recognizes all of the newly defined devices in the `devices.table` file.

For detailed descriptions of the `broadcast_file` and `configure_devices` commands, see the manual *OpenVOS System Administration: Configuring a System* (R287).

## Issuing the `configure_comm_protocol` Command

You must issue the `configure_comm_protocol` command to configure the vterm driver that the MST server process uses. The MST server process uses the vterm driver to attach TELNET client requests to operating system processes. Issue the following `configure_comm_protocol` command during the current bootload.

```
configure_comm_protocol vterm
```

See [“Manually Loading and Unloading SDLMUX” on page 3-23](#) for information about other `configure_comm_protocol` command lines that you can issue for STCP. See [“Modifying the `module\_start\_up.cm` and `start\_stcp.cm` Files” on page 6-3](#) for information about other commands that STCP requires in the `module_start_up.cm` file. See the manual *OpenVOS System Administration: Configuring a System* (R287) for more information about the `configure_comm_protocol` command.

## Confirming Device Configuration

To determine whether you have properly configured the vterm devices, issue the `list_devices` command. The `list_devices` command lists the path names of devices of a specified type. To confirm that the vterm devices are configured properly, issue the following command.

```
list_devices -type vterm
```

For a detailed description of the `list_devices` command, see the *OpenVOS Commands Reference Manual* (R098).

## Starting the MST Server Process Automatically

You can start the MST server process automatically by issuing the `telnet_msd` command from within the `start_stcp.cm` file and by issuing the `configure_comm_protocol vterm` command from within the `module_start_up.cm` file.

For information about issuing the `telnet_msd` command from within the `start_stcp.cm` file, see the [Explanation](#) in the `telnet_msd` command description.

To issue the `configure_comm_protocol vterm` command from within the `module_start_up.cm` file, uncomment the command line (if it exists) in your `module_start_up.cm` file, or add a new command line to the file. To uncomment a command line, delete the ampersand (&) and the space at the beginning of the command line. If the command line does not exist as a comment, add the following command line, which loads the vterm driver as a communications protocol on the module. Place this command line near the other `configure_comm_protocol` command lines.

```
configure_comm_protocol vterm
```

## The `telnet_msd` Command

This section describes the `telnet_msd` command, which you must issue to start the MST server process.

## telnet\_msd

***Privileged***

### Purpose

The `telnet_msd` command starts an MST server process that receives incoming TELNET connection requests. The MST server process then establishes a TELNET session, passes input and output between a TELNET client and a OpenVOS process, and closes the session. This command is typically invoked by a `start_process` command in the `start_stcp.cm` file.

### Display Form

```
----- telnet_msd -----
-network_port:      24
-max_sessions:     28
-error_severity:    2
-separate_log:      yes
-log_dir:           >system>stcp>logs
-vterm_sturname:    telnet*
-vterm_login:       yes
-vterm_slave_id:
-extension:         133
-force_edit:        yes
-EC_decimal_value:  8
-EL_decimal_value:  21
-local_ip:          no
-tcpwrapper_check:  no
-numeric:           no
```

## Command-Line Form

```
telnet_msd [-network_port port_num]
           [-max_sessions max_sessions]
           [-error_severity level_num]
           [-no_separate_log]
           [-log_dir dir_path]
           [-vterm_staname device_name]
           [-no_vterm_login]
           [-vterm_slave_id slave_id]
           [-extension x25_ext]
           [-no_force_edit]
           [-EC_decimal_value ec_char]
           [-EL_decimal_value el_char]
           [-local_ip]
           [-tcpwrapper_check]
           [-numeric]
```

## Arguments

- `-network_port port_num`

Specifies the port number at which the MST server process listens for incoming TELNET connection requests. The default value of `port_num` is 24.

### NOTICE

The standard STCP TELNET server process (`telnetd`) and the MST server process (`telnet_msd`) must not be listening for requests at the same network port. If you are running both of these server processes and you execute both the `telnet` command and the `telnet_msd` command, you must specify a different value for the port argument of each command (`-network_port` for the `telnet_msd` command and `port_number` for the `telnet` command).

- `-max_sessions max_sessions`

Specifies the maximum number of TELNET sessions that the MST server process can run concurrently. The value can be any number from 1 to 256. The default value is 28. See the [Explanation](#) for information about when you should change this value.

► `-error_severity level_num`

Specifies the severity level of status messages that are written to the log file. The MST server process writes messages of both the specified severity level and higher severity levels to the log file. The default severity level is 2. [Table F-1](#) lists the severity levels.

**Table F-1. Severity Levels of MST Status Messages**

Severity Level	Message Type	Description
1	INFO	Option negotiation messages
2	WARN	Nonfatal protocol errors
3	ERROR	Error messages from system functions
4	LOG	Operational messages (for example, Accept or Terminate)
5	FATAL	Fatal process errors

► `-no_separate_log`

**CYCLE**

Writes log data to the `mst_server_name.out` file instead of to a separate log file. By default (the value `yes`), the MST server process writes log data to the log file in the directory specified by the `-log_dir` argument. A new log file is created every day at one minute after midnight.

► `-log_dir dir_path`

Specifies the directory to which separate log files are written when the `-separate_log` argument is set to `yes`. The default directory path is `(master_disk)>system>stcp>logs`. This directory path is used automatically unless you specify a different path name. The name of each log file has the form `mst_log.date.N`, where `date` is the date on which the file is created (for example, 04-12-29) and `N` is a sequence number used to avoid duplicate names. A new log file is created every day at one minute after midnight.

► `-vterm_sturname device_name`

Specifies the name(s) of the vterm device(s) that the MST server process uses to establish a TELNET session for an OpenVOS process. Typically, the value of `device_name` is a prefix. The default value of `device_name` is `telnet*`. The star name must match one or more vterm device names in the `devices.tin` file. If no names match or if the vterm devices are already attached to other OpenVOS processes, the TELNET session is not established.

► `-no_vterm_login`

**CYCLE**

Selects a slave vterm device whose name matches the name specified with the `-vterm_sturname` argument. Slave vterm devices are defined in the `devices.tin` file by setting the `login_slave` field to 0. The possible values are

yes and no; the default value is *yes*, which corresponds to the value 1 in the *login\_slave* field. If you specify the default value, the MST server process selects login vterm devices.

#### NOTE

If you want to support both login and slave vterm devices, you must add to the *start\_stcp.cm* file a second *start\_process telnet\_msd* command line that specifies the argument *-no\_vterm\_login*. In this command line, you must also specify an appropriate port number with the *-network\_port* argument.

- ▶ *-vterm\_slave\_id slave\_id*  
Selects a slave vterm device. The default value is blank, which selects any one of the available slave vterm devices. If you specify a *slave\_id* value for this argument, the MST server process selects a slave vterm device whose *devices.tin* entry has a matching *slave\_id* value in the *parameters* field. (See the discussion of the *parameters* field in “[Fields Used in Entries for TELNET Vterm Devices](#)” on page F-3.) If you specify this argument, you must also specify the *-vterm\_sturname* and *-no\_vterm\_login* arguments.
- ▶ *-extension x25\_ext*  
Specifies the X.25 extension that the MST server process uses to establish a TELNET session. The MST server process generally performs the same function as the *x25\_exchange* process. It listens to the specified extension, calls that extension, and attaches a vterm device to establish the session. The default value is 133, which you should not need to change unless other processes (or applications) are running on the module that uses this extension.
- ▶ *-no\_force\_edit* CYCLE  
Specifies that the MST server process recognizes only standard TELNET erase requests when the client is in TELNET input mode; it will not recognize control sequences for the erase-character and erase-line requests. By default (the value *yes*), the MST server process recognizes the control sequences specified for the arguments *-EC\_decimal\_value* and *-EL\_decimal\_value*.
- ▶ *-EC\_decimal\_value ec\_char*  
Specifies the control character that a TELNET client can use to send the erase-character request. The value of *ec\_char* is the decimal value of the ASCII character that the MST server process recognizes as the erase-character request, unless you specified the *-no\_force\_edit* argument. The default value, 8, specifies a CTRL-H request, which performs a backspace on Stratus terminals.

- ▶ `-EL_decimal_value el_char`  
Specifies the control character that a TELNET client can use to send the erase-line request. The value of `el_char` is the decimal value of the ASCII character that the MST server process recognizes as the erase-line request, unless you specified the `-no_force_edit` argument. The default value, 21, specifies a `CTRL-U` request, which erases a line.
- ▶ `-local_ip` CYCLE  
Causes `telnet_msd` to bind only to the localhost IP address on the OpenVOS module. You can then use SSH tunneling to allow secure access.
- ▶ `-tcpwrapper_check` CYCLE  
Enables the MST server to authenticate each client that requests a TELNET connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files. As part of authenticating client requests, the server logs messages to the log files `tcpdallow` and `tcpddeny`. To enable authentication, specify the value `yes`. By default (the value `no`), the MST server does not authenticate clients and does not log messages in the log files.
- ▶ `-numeric` CYCLE  
Enables the MST server to check only the client's IP address and to ignore the host name when the server authenticates client access. By checking only the numeric IP address and ignoring the host name, the server can authenticate the client more quickly. The value that you specify for the `-numeric` argument takes effect only when the value of the `-tcpwrapper_check` argument is `yes`.

To enable the server to check the host name, specify the value `no` for the `-numeric` argument when you specify the value `yes` for the `-tcpwrapper_check` argument. If you specify the value `no` for the `-tcpwrapper_check` argument, the `-numeric` argument has no effect.

## Explanation

You can issue the `telnet_msd` command from the command line, though typically this command is invoked by a `start_process` command in the `start_stcp.cm` file. If you want the `start_stcp.cm` file to start an MST server process, you must edit the `start_stcp.cm` file to add a `start_process` command line with the `telnet_msd` command and use the `-network_port` argument to specify the port number at which the MST server process listens for incoming TELNET requests.

You stop the MST server process using the `stop_process` command.

The `telnet_msd` command starts an MST server process that can support one or more TELNET sessions, with the maximum number of sessions specified by the `-max_sessions` argument. Once the number of sessions has reached the number specified by the `-max_sessions` argument, the MST server process accepts no more connections until one of the existing connections is terminated. If an MST server



process is handling too many sessions, the performance of the individual sessions degrades. To improve performance, you must start multiple MST server processes, each listening to a different port, and tell users to use a different port if they cannot establish a connection using one port. An acceptable value for the `-max_sessions` argument depends upon the applications that users run. An MST server process can handle more keyboard-driven, interactive sessions than sessions that send large amounts of data to a user's terminal with little or no interaction from the user.

If you have a single-processor module, you can specify a greater value for the `-max_sessions` argument so that an MST server process can support more than 28 TELNET connections. There is no advantage to running multiple MST server processes on a single-processor module.

An MST server process writes status messages of both the severity level specified by the `-error_severity` argument and of higher severity levels to a log file in the directory specified by the `-log_dir` argument.

If you specify the `-no_separate_log` argument, the MST server process writes status messages to the `mst_server_name.out` file instead of to the directory specified by the `-log_dir` argument.

A new log file is created every day at one minute after midnight. Old log files are not deleted automatically; you must delete unneeded log files.

The MST server supports TCP wrappers authentication using the `-tcpwrapper_check` argument. If you specify the value `yes` for the `-tcpwrapper_check` argument, the MST server authenticates each client that requests a TELNET connection by checking the client's access as configured in the `hosts.allow` and `hosts.deny` files.

## NOTES

---

1. If you configure TCP wrappers functionality on your STCP server, a client may experience a small delay in TELNET services.
2. If the log files `tcpdallow` and `tcpddeny` become larger than 254 blocks, OpenVOS renames the files to `tcpdallow.yy-mm-dd.hh-mm` and `tcpddeny.yy-mm-dd.hh-mm` and starts new log files. (If you want to change the size at which log files become too large, contact the CAC.)

## Examples

To start the process from command level and to create the `telnet_msd.out` log file, issue the following command lines. You can also add these command lines to the `start_stcp.cm` file.

```
create_file (master_disk)>system>stcp>logs>telnet_msd.out

set_implicit_locking (master_disk)>system>stcp>logs>telnet_msd.out

start_process -output_path (master_disk)>system>stcp>logs>telnet_msd.out
+'(master_disk)>system>stcp>command_library>telnet_msd -max_sessions 28'
+-privileged -process_name telnet_msd
```

The following command lines provide another example.

```
create_file (master_disk)>system>stcp>logs>telnet_msd.out

set_implicit_locking (master_disk)>system>stcp>logs>telnet_msd.out

start_process -output_path (master_disk)>system>stcp>logs>telnet_msd.out
+'(master_disk)>system>stcp>command_library>telnet_msd -network_port 2999
+-vterm_staname slave_vterm -no_vterm_login' -privileged -process_name
+ telnet_msd
```

---

## Appendix G

# IPsec Support

Internet Protocol Security (IPsec) is a network-layer protocol standard that protects and authenticates IP packets. IPsec implemented at the network layer provides security for upper-layer protocols such as the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and the Internet Control Message Protocol (ICMP). IPsec can protect from alteration, observation, and spoofing of packets that any TCP/IP application transmits over the network. Hosts that want to communicate securely using IPsec must exchange an encryption key. The Internet Key Exchange (IKE) protocol handles the automatic distribution of a secure encryption key.

OpenVOS supports the strongSwan for OpenVOS implementation of IPsec. This implementation, which is based on strongSwan Release 5.9.0, replaces the IPsec implementation in earlier OpenVOS releases and is installed during the OpenVOS installation. You do not install strongSwan for OpenVOS as a separate product.

### NOTICE

---

Before using this product, contact the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative to have the purchase bit (S149) turned on, at no charge.

This appendix discusses the following topics:

- [“User Documentation” on page G-2](#)
- [“Release Contents” on page G-2](#)
- [“Software Requirements” on page G-2](#)
- [“Installing the Software” on page G-3](#)
- [“Configuration Files” on page G-4](#)
- [“Using the strongSwan man Pages” on page G-4](#)
- [“Starting and Stopping the strongSwan Daemon” on page G-4](#)
- [“Supported Algorithms” on page G-5](#)
- [“Security Policies” on page G-6](#)

- “Avoiding Problems When Tunneling” on page G-9
- “Important Considerations” on page G-10

## User Documentation

This appendix provides the information needed to use strongSwan for OpenVOS. For user information about the standard strongSwan implementation, see <https://wiki.strongswan.org/projects/strongswan/wiki/UserDocumentation>.

### NOTICE

Be sure to read the strongSwan documentation, including the man pages for `ipsec.conf` and `ipsec.secrets`, before attempting to use IPsec. If you are unfamiliar with the concepts of encryption, IPsec, and certificates, you should learn about these concepts with an online course or reference manuals before attempting to read and use the strongSwan documentation.

## Release Contents

strongSwan for OpenVOS is installed in the `(master_disk)>system>strongswan` directory.

Add `(master_disk)>system>strongswan>sbin` to your command library path so that you can run the strongSwan commands.

### NOTE

The `(master_disk)>system>strongswan` directory contains many more directories and files than this appendix describes. For more information about these directories and files, see the website listed in “User Documentation” on page G-2.

## Software Requirements

This implementation of strongSwan for OpenVOS requires the following software:

- OpenVOS Release 19.3.1 or later
- OpenVOS GNU Tools Release 3.5 or later is also required in order to run strongSwan commands using the `ipsec` command.

## Installing the Software

When you issue the `install_new_release` command to install OpenVOS, the command also installs strongSwan for OpenVOS in the following directories. For the sake of simplicity, this list does not include all subdirectories of the directories shown here.

```
(master_disk) >sytem>strongswan
(master_disk) >system>strongswan>etc
(master_disk) >system>strongswan>etc>ipsec.d
(master_disk) >system>strongswan>etc>ipsec.d>private
(master_disk) >system>strongswan>etc>ipsec.d>certs
(master_disk) >system>strongswan>etc>ipsec.d>crls
(master_disk) >system>strongswan>etc>ipsec.d>cacerts
(master_disk) >system>strongswan>etc>ipsec.d>ocspcerts
(master_disk) >system>strongswan>etc>ipsec.d>aacerts
(master_disk) >system>strongswan>etc>ipsec.d>acerts
(master_disk) >system>strongswan>etc>ipsec.d>reqs
(master_disk) >system>strongswan>etc>strongswan.d
(master_disk) >system>strongswan>etc>strongswan.d>charon
(master_disk) >system>strongswan>etc>swanctl
(master_disk) >system>strongswan>etc>templates
(master_disk) >system>strongswan>bin
(master_disk) >system>strongswan>lib
(master_disk) >system>strongswan>lib>ipsec
(master_disk) >system>strongswan>lib>ipsec>plugins
(master_disk) >system>strongswan>libexec
(master_disk) >system>strongswan>libexec>ipsec
(master_disk) >system>strongswan>sbin
(master_disk) >system>strongswan>logs
(master_disk) >system>strongswan>share>man
(master_disk) >system>strongswan>share>strongswan
(master_disk) >system>strongswan>var
```

It also installs the following files:

```
(master_disk) >system>strongswan>bin>pki.pm
(master_disk) >system>strongswan>libexec>ipsec>_copyright.pm
(master_disk) >system>strongswan>libexec>ipsec>charon.pm
(master_disk) >system>strongswan>libexec>ipsec>scepclient.pm
(master_disk) >system>strongswan>libexec>ipsec>starter.pm
(master_disk) >system>strongswan>libexec>ipsec>stroke.pm
(master_disk) >system>strongswan>sbin>ipsec
(master_disk) >system>strongswan>sbin>swanctl1.pm
```

## Configuration Files

strongSwan for OpenVOS uses the following configuration files, all of which are located in the `(master_disk)>system>strongswan>share>strongswan>templates` directory:

- `ipsec.conf`—Provides the configuration of IPsec connections.
- `ipsec.secrets`—Lists the secrets (for example, pre-shared keys and private keys).
- `ipsec.d`—Stores certificates and private keys.
- `strongswan.conf`—Allows you to configure global settings.

### NOTE

You must create all of the configuration files, including `ipsec.secrets`, as stream or 64-bit stream files, if you do not copy them from the template file.

For more information about the configuration files, see the website listed in [“User Documentation” on page G-2](#).

## Using the strongSwan man Pages

The strongSwan man pages are located in the `(master_disk)>system>strongswan>share>man` directory.

To access the strongSwan man pages, use the `man.pm` command that is provided with OpenVOS GNU Tools. The `man.pm` command is located in the `(master_disk)>system>gnu_library>bin` directory.

To view the strongSwan man pages, you must edit the `(master_disk)>system>gnu_library>etc>man.conf` file to include the following MANPATH entry for the supplied man pages:

```
MANPATH    /system/strongswan/share/man
```

## Starting and Stopping the strongSwan Daemon

This section discusses the following topics:

- [“Starting the strongSwan Daemon During Module Startup” on page G-5](#)
- [“Starting the strongSwan Daemon from a Running Module” on page G-5](#)
- [“Stopping the strongSwan Daemon” on page G-5](#)

## Starting the strongSwan Daemon During Module Startup

To start the strongSwan daemon during module startup, make sure that the following command lines in the `start_stcp.cm` file appear and are uncommented:

```
& if (exists strongswan.out)
& then !rename strongswan.out strongswan.&log_suffix&.out -delete
& if (command_status) ^= 0
& then &goto other_errors
!create_file strongswan.out
& if (command_status) ^= 0
& then &goto other_errors
!set_implicit_locking strongswan.out

& *****STRONGSWAN*****
&
& !display_line Starting strongswan...
& !start_process
(master_disk)>system>strongswan>libexec>ipsec>starter.pm -root
+ -privileged -process_name strongswan -output_path
(master_disk)>system>stcp>l
+ogs>strongswan.out
& !display_line STRONGSWAN .....
```

## Starting the strongSwan Daemon from a Running Module

You can control the strongSwan daemon with the `ipsec` command, which is installed in the `(master_disk)>system>strongswan>sbin` directory as a shell script. Issue the `ipsec start` command to start the starter daemon, which, in turn, starts and configures the keying daemon `charon`. The process must start as `root`.

## Stopping the strongSwan Daemon

Issue the `ipsec stop` command to stop the starter daemon, which, in turn, stops the keying daemon `charon`.

## Supported Algorithms

This section discusses the following topics:

- [“Supported Integrity Algorithms for AH and ESP” on page G-6](#)
- [“Supported Encryption Algorithms for ESP” on page G-6](#)
- [“Algorithm Performance” on page G-6](#)

### Supported Integrity Algorithms for AH and ESP

The OpenVOS Release 19.3.1 kernel supports the sha, sha1, sha256, sha2\_256, sha384, sha2\_384, sha512, sha2\_512, aes128gmac, aes192gmac, and aes256gmac integrity algorithms.

### Supported Encryption Algorithms for ESP

The OpenVOS Release 19.3.1 kernel supports the aes, aes128, aes192, aes256, aes128ctr, aes192ctr, aes256ctr, aes128gcm16, aes128gcm128, aes192gcm16, aes192gcm128, aes256gcm16, and aes256gcm128 encryption algorithms. The AES/GCM selections perform encryption and integrity checking.

### Algorithm Performance

The AES and GMAC computations have support for hardware acceleration. The GCM encryption/integrity selection (that is, AES combined with GMAC) is roughly an order of magnitude faster than any other combination of encryption/integrity.

## Security Policies

This section discusses the following topics:

- “Default Security Policy” on page G-6
- “Ordering of Security Policies” on page G-9

### Default Security Policy

The ipsec.conf file for strongSwan defines security policies for combinations of protocols, addresses, and ports. The strongSwan documentation (see “User Documentation” on page G-2) describes the format of this file.

In addition, a default security policy applies to any combination of protocol, addresses, and ports that is not defined in ipsec.conf. You can set the default security policy with the access\_info\_monitor command, using the OID 1.3.6.1.4.1.458.114.1.6.1.3.6.1.2.1.4.119, ipVosDefaultIPsecPolicy. You can use the following values:

Value	Policy	Description
0	discard	Packets not matched by the definitions in ipsec.conf are discarded unless they are required for proper operation of the Internet protocol. See Table G-1 for a list of the required packet types.
1	passthru	Packets not matched by the definitions in ipsec.conf are sent or received.



The `discard` default value is the most secure but can easily make the module inaccessible through the network. You should define an emergency connection at the beginning of the `ipsec.conf` file (for example, `ssh` from a specific location) to allow connection if `ipsec.conf` contains configuration errors.

**Table G-1** describes packet types that are required and/or useful for proper Internet operation. Required packets are still passed through by the `discard` default security policy. If you configure these packets in `ipsec.conf`, be sure to give the packet type as the source port.

**Table G-1. Packet Types Needed for Internet Communication** (Page 1 of 2)

Protocol	Packet Type	Required?	Description
icmp (1)	3 - Unreachable	Yes	Sent by a peer or router to indicate that a packet cannot be delivered for various reasons. If this is blocked, applications trying to use incorrect addresses and/or ports hang for very long periods of time.
	5 - Redirect	Yes	Sent by routers to provide a more efficient transmission path.
	11 - Time Exceeded	Yes	Sent by a peer when it cannot reassemble fragmented packets. Sent by routers if the time to live is exceeded when forwarding packets (usually, this indicates a routing loop).
	12 - Param Problem	Yes	Sent when a peer receives an incorrectly formatted packet.
	0 - Echo Reply	No	Pings replies.
	8 - Echo	No	Pings. Useful for diagnosing network connectivity issues. These can present a security problem because they can be used to probe a network.
igmp (2)	All	Yes	Configures the reception and forwarding of IPv4 multi-cast addresses.
ARP	All	N/A	ARP is not part of the IP protocol suite; therefore, it is not affected by IPsec.

**Table G-1. Packet Types Needed for Internet Communication** (Page 2 of 2)

Protocol	Packet Type	Required?	Description
icmpv6 (58)	1 - Dest Unreach	Yes	Sent by a peer or router to indicate that a packet cannot be delivered for various reasons. If this is blocked, applications trying to use incorrect addresses and/or ports hang for very long periods of time.
	2 - Packet Too Big	Yes	Sent by a peer or router to indicate that a packet is too large to be routed. Required by IPv6.
	3 - Time Exceeded	Yes	Sent by a peer when it cannot reassemble fragmented packets. Sent by routers if the time to live is exceeded when forwarding packets (usually, this indicates a routing loop).
	4 - Param Problem	Yes	Sent when a peer receives an incorrectly formatted packet.
	130 - MLD Listener Query	Yes	Configures reception and forwarding of IPv6 multi-cast addresses.
	131 - MLD Listener Report	Yes	Configures reception and forwarding of IPv6 multi-cast addresses.
	132 - MLD Listener Reduction	Yes	Configures reception and forwarding of IPv6 multi-cast addresses.
	133 - ND Router Solicit	Yes	Requests information from any routers on a subnet.
	134 - ND Router Advertisement	Yes	Used by all peers to find routers. There is no default route for IPv6; therefore, this is the only way a host knows how to send packets off the local subnet. Can also be used for automatic IPv6 address assignment.
	135 - ND Neighbor Solicit	Yes	Finds the MAC address for an IPv6 address. This is the IPv6 equivalent of ARP.
	136 - ND Neighbor Advert	Yes	Finds the MAC address for an IPv6 address. This is the IPv6 equivalent of ARP.
	137 - ND Redirect	Yes	Sent by routers to provide a more efficient transmission path.
	128 - Echo Request	No	Pings. Useful for diagnosing network connectivity issues. These can present a security problem because they can be used to probe a network.
	129 - Echo Reply	No	Pings replies.

## Ordering of Security Policies

The IPsec standards require that security policies (also known as *configurations*) are searched in order. Security policies are ordered by priority (lowest to highest), not by the order in which the configurations appear in the `ipsec.conf` file. strongSwan for OpenVOS computes the priority based on the following factors:

- A base priority is assigned as follows: 200,000 for bypass policies, 400,000 for IPsec policies, and 600,000 for drop policies.
- The subnet mask length multiplied by 512 is subtracted for each address. In other words, policies with longer subnet masks are ordered first.
- 256 is subtracted if a protocol is specified.
- 64 is subtracted if a source port is specified.
- 64 is subtracted if a destination port is specified.
- 1 is subtracted if the policy is added dynamically (that is, if it is not included in the `ipsec.conf` file).

To view the actual ordering of security policies, use the `dump_ipsec -no_sadb` request in `analyze_system`. See *OpenVOS System Analysis Manual (R073)* for more information about this request.

## Avoiding Problems When Tunneling

strongSwan requires a source and destination address in order to look up a tunnel. Therefore, the source address must be defaulted before applying strongSwan. Unfortunately, for a packet that will be tunneled, a local interface typically does not exist, and you receive an error such as `EHOSTUNREACH` when strongSwan tries to compute a default source.

If the tunnel and the packet being tunneled are both IPv4 or IPv6, you can use `auto=route` with the connection definition. strongSwan then creates a route to enable default source addresses to be created.

If a tunnel and the packet being tunneled are not the same IP version, you must add the route manually.

- The source of the route should be the subnet from the remote end of the tunnel.
- The destination of the route should be the IP address of an interface that is in the subnet for the local end of the tunnel.

You can add a route by using the `route` command.

## Important Considerations

Note the following important considerations related to strongSwan for OpenVOS:

- By default, the IP forwarding feature is off for IPv6 and explicitly disabled for IPv4 in `start_stcp.cm`. If you use IPsec tunneling, turn on IP forwarding for IPv4 in `start_stcp.cm` as follows:

```
(master_disk)>system>stcp>command_library>IP_forwarding on
```

Turn on IP forwarding for IPv6 in `start_stcp.cm` as follows:

```
access_info_monitor 1.3.6.1.2.1.4.25 set 1
```

- Do not use the `.der` encoded certificate format to communicate between the OpenVOS Release 18.x and later IPsec implementation and the OpenVOS Release 17.2.x IPsec implementation. Instead, use the `.pem` encoded certificate format. You can, however, use the `.der` format for certificates located on the OpenVOS Release 18.x and later IPsec implementation.
- As of Release 19.3.1, OpenVOS does not support the `charon-systemd` daemon or the `vici` control interface. Therefore, you must continue to use the previous tools for starting strongSwan and for controlling the daemon interface.
- Be sure to read the strongSwan documentation for the `lifetime`, `ikelifetime`, `marginetime`, `rekeyfuzz`, `lifebytes`, and `marginbytes` connection parameters. The `lifebytes` and `marginbytes` parameters do not have default values but must be configured for the strongSwan for OpenVOS implementation of IPsec.
  - Set the `lifebytes` parameter so that rekeying occurs before enough data has been transmitted to compromise the security of the key. This limit depends on the encryption algorithm that is used as well as the desired security level.
  - Set the `marginbytes` parameter to a value that is large enough to be at least 3 minutes at the highest possible transmission rate. If you set the value too low, errors occur when strongSwan rekeys.
- The kernel sets the values of the `lifepackets` and `marginpackets` parameters; you cannot change these values.
- If you are seeing TCP packet loss or other network issues related to lost packets, use the `netstat` command to check the IP parameters to see if the packet loss is occurring because of the replay window. If it is, you can use the `replay_window` connection parameter to increase the window size, or you can turn off replay-window checking by setting `replay_window` to zero. However, turning off replay-window checking is not recommended.

---

# Index

## A

### Access lists

- access control lists, 8-8
- device, 4-30

### Access, restricting

- `hosts.allow` database file, 5-10
- `hosts.deny` database file, 5-10
- TCP wrappers functionality, 5-10
- to files, 8-9, 8-16
- to STREAMS drivers and modules, 4-30

`access_info_monitor` command, 9-4

Access-layer device driver, TELNET, 4-8, 4-9, 4-10

`action` argument of the `dlmux_admin` command, 9-16

Active network interface, 9-16, 9-17, 9-18

Adapters. *See* Ethernet PCI adapters

`add_partner` action of the `dlmux_admin` command, 9-16

`add_static_route` subcommand, 9-110

### Adding

- a partner, 3-29, 9-16, 9-19
- a simplex Ethernet PCI adapter, 3-30
- two new network interfaces as partners, 3-6

### Address Resolution Protocol (ARP)

- defined, 1-8
- deleting and setting table entries, 9-11
- driver, 1-8
- response packets, 9-12

ADP driver, 1-8

AF UNIX driver, 1-8, 4-1, 4-7

- device entry, 4-7

### Aliases

- host name, 5-9
- network, 5-27
- protocol, 5-41
- service, 5-48

`analyze_system` command

`dump_sdlnmux` request, 3-31, 9-18

`list_stcp_params` request, 6-21, 7-1, 7-4

`set_stcp_param` request, 6-21, 7-1

`stcp_meters` request, 6-21

Application programming interface (API), 1-6

Applications supported by STCP, 1-2

AREA entry in `ospfdconf` file, 5-33

Area ID, OSPF router interface, 5-34

Areas, OSPF, 5-32

`arp` command, 1-9, 9-10

arguments, 9-11, 9-171

examples, 9-12

ARP. *See* Address Resolution Protocol (ARP)

AS. *See* Autonomous system (AS)

Attaching applications to TELNET slave devices, C-1

AUTH entry in `ospfdconf` file, 5-34

### Authentication

`ftpd` daemon process, 8-13

`hosts.allow` database file, 5-10

`hosts.deny` database file, 5-10

OSPF, A-21

`snmpd` daemon process, 8-30

TCP wrappers functionality, 5-10

`tcpd` daemon process, 8-32

`tftp` command, 9-171

`tftpd` daemon process, 8-42

Automatic hardware failover. *See* Hardware failover

Autonomous system (AS), 8-24, A-19

## B

Backbone area, OSPF

virtual links, 5-36, A-20

Backup designated router (BDR), 5-35, A-19

**BOOTP**

- configuration files directory, 5-5
- configuring, 5-5
- TCP wrappers functionality, 8-5
- bootpd daemon process, 8-4
  - bootplog log file, 8-5
  - inetd.conf database file, 8-4, 8-5
  - services database file, 8-6
  - TCP wrappers functionality, 8-5
- bootpd.pm program module, 1-9
- bootplog log file, 8-5
- bootptab database file, 1-13
  - described, 5-4
  - editing, 5-5
  - fields, 5-5
- Border router, defined, A-19
- broadcast\_file command, 6-11

**C**

- Changing SDLMUX configurations, 3-26
- Character-generator (chargen) service, 5-22
- Checking status of partnered network
  - interfaces, 9-16, 9-17, 9-18
- Child processes, FTP, 1-10, 8-11
- Client-initiated TELNET connections, 4-20,  
4-26, 4-28, C-1
- Clonable device limit
  - AF UNIX driver, 4-8
  - IP drivers, 4-6
  - login devices, 4-20
  - loopback driver, 4-7
  - slave devices, 4-26, 4-28
  - STCP protocol drivers, 4-2, 4-3
- Clonable device name, 4-12, 4-20, 4-21, 4-25,  
5-56, C-2, C-4, C-8
- Clonable drivers
  - configuring, 4-2
- clone\_limit field, devices.tin file, 4-3,  
4-5, 4-6, 4-7, 4-8, 4-20, 4-26, 4-28
- clone\_limit\_32 field, devices.tin  
file, 4-4, 4-28
- Command library, STCP, 1-6
- Command macros, 1-6
  - mddmon\_start\_up.cm, 6-7
  - module\_start\_up.cm, 6-3
  - starting inetd (start\_inetd.cm), 1-6,  
1-14, 2-4, 6-1, 6-10, 6-11, 8-20

- starting STCP (start\_stcp.cm), 1-6,  
2-3, 6-4
- starting xinetd  
(start\_xinetd.cm), 1-14, 6-10
- stopping inetd (stop\_inetd.cm), 1-6,  
1-14, 6-2, 8-21
- stopping xinetd  
(stop\_xinetd.cm), 1-14

**Commands**

*See also specific command names*

- access\_info\_monitor, 9-4
- analyze\_system, 9-18
- arp, 9-10
- brief descriptions, 9-2
- dlnux\_admin, 3-10, 3-21, 9-15
  - error messages, 9-22
  - examples, 9-20
  - initializing Ethernet PCI  
adapters, 3-10, 3-21
- ftp, 9-25
- hostname, 9-35
- ifconfig, 3-10, 3-25, 9-37
- IP\_forwarding, 9-53
- ip\_pair\_admin, 9-55
- ip6addrctl, 9-60
- list\_devices
  - Ethernet PCI adapters, 3-21
  - STREAMS devices, 3-20
- list\_kernel\_programs, 3-25, 6-13
- load\_kernel\_program, 3-24
- netstat, 9-61
- ngrep, 9-106
- omon, 9-107
- overview, 9-1
- packet\_monitor, 9-117
- ping, 9-129
- route, 9-140
- rtsol, 9-154
- set\_udp\_ordering, 9-155
- setkey, 9-157
- tcpdump, 9-158
- telnet, 9-159
- telnet\_admin, 9-163
- tftp, 9-170
- tftp6, 9-173
- traceroute, 9-174

**Communications adapters, 1-15**

*See also Ethernet PCI adapters*

Communities of the Simple Network  
 Management Protocol (SNMP), 8-30

Configuration checklist, 2-1

- `ospfd` daemon process, 2-7, 2-9
- STCP, 2-2
- TCP wrappers functionality, 2-7
- `telnetd` daemon process, 2-5

Configuration requirements

- partnered network interfaces, 3-4, 3-6, 3-10
- SDLMUX, 3-3
  - example, 3-11
- simplex network interfaces, 3-4, 3-6

Configuration-table files, `devices.table`, 3-1

`configure_devices` command, 3-2

Configuring SDLMUX devices, 3-11, 3-12

Connections

- FTP, 9-25
- TCP protocol states, 9-71
- TELNET, 9-159, 9-163

Converting a simplex Ethernet PCI adapter to a partnered one, 3-29

`create_table` command, 3-2, 4-30, 6-11

## D

Daemon processes

- See *also* individual daemon process names
- brief descriptions, 6-2, 8-1
- overview, 8-1
- verifying, 6-16

Data transfer, enabling and disabling on a network interface, 9-39

Database files, 5-1

- templates described, 1-13
- templates directory, 1-6

Data-Link Multiplexer, 1-9

- See *also* SDLMUX

Date and time stamp on log files, 6-8, 8-11

daytime service, 5-22

Debug logging, OSPF, 9-110

Defining network interfaces, 3-6, 3-25

`delete_partner` action of the `dlmux_admin` command, 9-16

Deleting

- a partner, 3-28, 9-16, 9-19
- an SDLMUX-group interface, 3-27

Denial-of-Service attack, 7-2

Designated router (DR), A-19

- election process priority, 5-35

Destination file, 9-107

- template, 9-108

Destination number, 9-109

Destinations, listing OSPF routers, 9-109

Device access lists, 4-30

Device prefix, 4-12, 4-20, 4-21, 4-25, 5-56, C-2, C-4, C-8

Device types, 4-3, 4-10, 4-18, 6-17

- Ethernet PCI adapter, 3-15
- SDLMUX-group interfaces, 3-13
- streams, 3-13
- `streams_pci`, 3-15

`device_name` argument of the `dlmux_admin` command, 9-15

`device_type` field, `devices.tin` file, 4-3, 4-4, 4-5, 4-6, 4-7, 4-10, 4-11, 4-18, 4-25

Devices

- access lists, 4-30
- AF UNIX protocol driver, 4-7
- configuration process overview, 3-1
- confirming configuration of, 6-14
- partnered, 3-6, 3-16, 3-18, 3-25
- simplex, 1-15, 3-6, 3-8
- STCP protocol drivers, 4-2
  - controlling access, 4-30
  - IP, 4-5
  - loopback, 4-6
  - STCP, 4-2
  - UDP, 4-4
- STREAMS, 4-1, 4-2, 6-17
- TELNET, 4-8
  - incoming slave, 4-20, 6-18, C-1
  - login, 4-12, 4-18, 6-18
  - outgoing slave, 4-26, 6-18, C-5
  - pipe entries, 4-11
  - TLI access-layer device driver, 4-10
- virtual terminal (vterm), F-2
  - login, F-5, F-11
  - MST server, F-2
  - slave, F-5, F-11

`devices.table` file, 3-1, 3-2, 4-30, 6-11

`devices.tin` file, 6-11

- AF UNIX driver, 4-7
- `clone_limit` field
  - Ethernet PCI adapters, 3-18
  - SDLMUX-group interfaces, 3-13

- device\_type field
  - streams, 3-13
  - streams\_pci, 3-15
- editing, 3-1
- Ethernet PCI adapters, 3-14
- fields
  - See also individual field names
  - device\_type, 3-13, 3-15
  - incoming slave devices, 4-25
  - loopback driver, 4-6
  - name, 3-14
  - outgoing slave devices, 4-28
  - parameters, 3-18
  - priv\_terminal, 4-13
- hardware\_id field for Ethernet PCI adapters, 3-15
- incoming slave devices, 4-25
- IP driver, 4-5
- module\_name field
  - Ethernet PCI adapters, 3-14
  - SDLMUX-group interfaces, 3-13
- name field
  - Ethernet PCI adapters, 3-14
  - SDLMUX-group interfaces, 3-13
- outgoing slave devices, 4-28
- parameters field, 3-16, 3-17
- partnered Ethernet PCI adapters, 3-16, 3-18
- protocol driver entries, 4-2
- sample, B-3
- SDLMUX-group interfaces, 3-12
- STCP driver, 4-3
- streams\_driver field
  - Ethernet PCI adapters, 3-15
  - igb, 3-15
  - ixgbe, 3-15
  - SDLMUX devices, 3-13
- TELNET entries, 4-8
- TELNET login devices, 4-17
- TELNET pipes, 4-11
- UDP driver, 4-4
- verifying, 3-20

diag\_nio\_monitor.pm program

- module, 6-7

diag\_ups\_monitor.pm program

- module, 6-7

dialup field, devices.tin file, 4-20

Directories ((master\_disk))

- >system>acl, 4-30
- >system, 3-2
- >system>configuration, 3-2, 4-30
- >system>kernel\_loadable\_library, 1-5
- >system>release\_number, 1-14
- >system>stcp, 1-5, 6-7
- >system>stcp>command\_library, 1-5, 6-3, 6-13
- >system>stcp>include\_library, 1-5
- >system>stcp>logs, 1-5
- >system>stcp>object\_library, 1-6
- >system>stcp>templates, 1-5, 2-7, 5-2, 5-31, 9-108

Discard service (discard\_stream.pm), 5-22

dlmux\_admin command, 1-17, 6-5, 6-11, 9-15

arguments

- action, 9-16

error messages, 9-22

examples, 9-20

initializing Ethernet PCI adapters, 3-10, 3-21

SDLMUX, 3-10

loading, 3-21

Domain Name Service (DNS), 5-7, A-23

servers, 5-43

Domains, A-23

DOS attack, 7-2

Drivers

- communications, 1-15
- IGB, 1-8, 1-16
- IXGBE, 1-8, 1-16
- protocol, configuring
  - AF UNIX, 4-7
  - IP, 4-5
  - loopback, 4-6
  - STCP, 4-2
  - UDP, 4-4
- SDLMUX, 1-16
- setting access to, 4-30
- STCP protocol
  - associating with device access lists, 4-30
  - confirming configuration of, 6-14
- TLI (TELNET), 4-10
- verifying, 6-15
- virtual terminal (vterm), F-6



Dual stack functionality, 1-3  
 dump\_sdlmux request of the  
     analyze\_system command, 3-31,  
     9-18  
 Duplex setting of PCI adapters, 3-17  
 Duplexed network interfaces. *See* Partnered  
 network interfaces

## E

Echo service (echo.stream.pm), 5-22  
 Embedded 10/100/1000-Mbps Ethernet PCI  
 Adapters  
     network interfaces, 3-4, 3-5  
     SDLMUX-group interfaces, 3-4, 3-5  
 enable interface subcommand, 9-110  
 Error messages  
     arp command, 9-14  
     dlmux\_admin command, 9-22  
     ftp command, 9-34  
     in log files, 8-2  
     netstat command, 9-63  
     OSPF, E-1  
     ping command, 9-135  
     telnet command, 9-162  
     tftp command, 9-172

## Errors

    indicated by netstat statistics, 9-95  
     resolving, 10-1

Establishing a logical network interface, 3-6

## Ethernet PCI adapters

    adding, 3-29, 3-30, 9-16  
     hardware failover, 1-15  
     initializing, 3-21, 9-17  
     logical configuration, 3-6  
     name, 3-14  
     network interfaces, 3-3  
     partnered, 1-15, 1-16  
     physical configuration, 3-3  
     SDLMUX, 1-15  
     SDLMUX-group interfaces, 3-6  
     simplex, 1-15  
     status, 9-16, 9-17, 9-18  
     uninitializing, 9-17

Event logging, OSPF, 9-110, 9-111, 10-6

event\_logging subcommand, 9-110

## Examples

    arp command, 9-12  
     bootptab database file, B-8

    devices.tin file, 3-13, B-3  
     dlmux\_admin command, 9-20  
     ftp command, 9-32  
     hostname command, 9-36  
     hosts database file, B-8  
     hosts.allow database file, 5-16  
     hosts.deny database file, 5-16  
     ifconfig command, 3-26, 9-49  
     incoming slave device entries, 4-22, 4-23  
     inetd.conf database file, B-9  
     IP\_forwarding command, 9-54  
     login device entries, 4-13, 4-14  
     networks database file, 5-27, B-9  
     nsswitch.conf database file, 5-29  
     omon subcommands, 9-114–9-115  
     ospfd entries in start\_stcp.cm  
         command macro, 8-25  
     ospfdconf database file, B-10  
     packet\_monitor command, 9-125  
     ping command, 9-135  
     protocols database file, 5-42  
     resolv.conf database file, 5-46  
     route command, 9-146  
     SDLMUX configuration, 3-11  
     services database file, 5-48, 9-169,  
         B-11, C-3  
     snmpconf database file, B-11  
     telnet command, 9-161  
     telnet\_admin command, 9-168  
     telnetd daemon process, C-4  
     telnetSERVICE database file, 4-13,  
         4-14, 4-22, 4-23, 5-56, 9-169,  
         B-12, C-3, C-4  
     tftp command, 9-172  
     traceroute command, 9-178  
     VLSN configuration, A-17

## F

failover action of the dlmux\_admin  
     command, 9-16

## Failover, hardware

*See also* Hardware failover  
     definition, 1-15  
     partnered network interfaces, A-12

## File Transfer Protocol (FTP)

    client, 9-25  
     daemon, 8-7  
     TCP wrappers functionality, 8-13

## Files

See *also* specific file names

`devices.tin`, 3-10, 3-12

`inetd` start-up, 1-6, 1-14, 2-4

`inetd`, stopping, 1-6, 1-14

log, 6-8, 8-2

machine-executable, 3-2, 4-30

module start-up, 6-3

`mst_server_name.out`, F-10

restricting access, 8-9, 8-16

STCP start-up, 1-6, 1-14, 6-7

STCP, stopping, 1-6

templates, 1-13, 5-2

transferring with `ftp` command, 9-25

transferring with `tftp` command, 9-170

`xinetd` start-up, 1-14

`xinetd`, stopping, 1-14

`flag` subcommand, 9-111

Forward broadcast (`forwb`) option, A-21

`ftp` command, 1-9, 9-2, 9-25

arguments, 9-25, 9-26

examples, 9-32

FTP. See File Transfer Protocol (FTP)

`ftpd` daemon process, 1-10, 8-7

arguments, 8-7, 9-5

`ftpd_ch.pm` program module, 1-10

`ftpd.out` log file, 8-7

`start_stcp.cm` command macro, 8-15

TCP wrappers functionality, 8-10

`ftpd_ch.pm` program module, 8-11

`ftpd.out` log file, 8-7, 8-8

## G

Gateways. See Routers/Gateways

`get_status` action of the `dldmux_admin`  
command, 9-16

## H

Hardware failover, A-12

definition, 1-15

requesting a manual failover, 9-16, 9-19

SDLMUX, 1-15

Hello

interval, 5-35

packets

described, A-19

interval, 5-35

Host addresses, A-4

HOST entry in `ospfdconf` file, 5-36

Host name

entry in destination file, 9-108

local

setting, 9-35

verifying, 6-13

Host route, in `ospfdconf` file, 5-36

`host_route_information`

subcommand, 9-111

Host-initiated TELNET connections, 4-26, 4-28,  
4-29, C-1, C-5, C-7

`hostname` command, 1-10, 6-11, 6-13, 9-35

arguments, 9-35

examples, 9-36

generated host file, 9-35

initializing STCP, 6-4

Hosts

adding to hosts database file, 5-9

`arp` command argument, 9-11

defined, A-1

`ftp` command argument, 9-2, 9-25

`hostname` command argument, 9-35

`packet_monitor` command

argument, 9-121

`ping` command argument, 9-130

`route` command argument, 9-141

`telnet` command argument, 9-159

`tftp` command argument, 9-170

`traceroute` command argument, 9-175

`hosts` database file, 1-13, 5-8, 9-10, 9-25,  
9-130, 9-141

defining hosts, A-4

described, 5-7

editing, 5-8

fields, 5-8

`hosts.allow` database file, 1-13

described, 5-10

editing, 5-12

example entries, 5-17

fields, 5-12, 5-13

`hosts.deny` database file, 1-13

described, 5-10

editing, 5-12

example entries, 5-17

fields, 5-12, 5-13

## I

- IF entry in `ospfdconf` file, 5-34, 5-35
- `ifconfig` command, 1-10, 6-21, 9-37
  - arguments, 9-39, 9-41
  - defining logical network interfaces, A-4
  - examples, 9-49
  - ignoring other arguments, 9-48
  - SDLMUX-group interfaces, 3-10, 3-25
  - specifying IP address, 9-39
  - `start_stcp.cm` command macro, 6-8
  - VLSN configuration, A-17
- IGB driver, 1-8, 1-16, 3-15
  - verifying configuration, 3-25, 6-14
- IGMP
  - reports, 9-9
  - specifying with `netstat`, 9-63, 9-67, 9-68
- Include library, 1-6
- Incoming slave devices
  - configuring, 4-8
  - device entries, 4-20
  - for incoming slave services, 5-53
- `inetd` daemon process, 1-10, 8-19
  - `inetd.conf` database file, 5-21, 8-21
  - process name `stcp_inetd`, 8-20
  - TCP wrappers functionality, 2-8, 5-21
  - `tcpd` daemon process, 5-21, 5-23, 8-32
- `inetd.conf` database file, 1-13, C-4
  - `bootpd` entries, 8-5
  - described, 5-21, 5-57
  - editing, 5-22
  - fields, 5-23
  - `services` database file, 5-21, 5-47
  - TCP wrappers functionality, 2-8, 5-21
  - `tcpd` daemon process, 8-32
  - `tftpd` entries, 8-5
- `inetd.out` log file, 8-19, 8-20
- `init_sdlnux` action of the `dlnux_admin` command, 9-17
- Initializing
  - Ethernet PCI adapters, 3-21, 9-17
  - SDLMUX-group interfaces, 3-21, 9-17
  - STCP, 6-11
- Interfaces
  - local host, 5-8
  - loopback, 5-8
  - OSPF routers
    - cost, 5-34
    - enabling and disabling, 9-110
    - in `ospfdconf` file, 5-32, 5-34
    - listing with `omon` command, 9-111
    - metric, 5-34
    - type, 5-34
  - Internet Assigned Numbers Authority (IANA), A-4
  - Internet Protocol (IP)
    - addresses
      - adding to a network interface, 6-21, 6-22, 9-43
      - `ipconfig` command argument, 9-39
      - OSPF router interfaces, 5-34
    - driver, 1-9
      - device configuration, 4-5
    - multicast, A-7
    - network classes, A-3
  - IP forwarding
    - default settings, 9-53
    - disabling, 9-53
    - enabling, 9-53
  - IP multicast interface, specifying an alias, 9-145
  - IP multicast transmission service, A-7
  - IP unicast interface, default, 9-145
  - `IP_address` entry in destination file, 9-108
  - `IP_forwarding` command, 1-10, 1-17, 1-18, 9-53
    - arguments, 9-53
    - examples, 9-54
  - `ip_pair_admin` command, 9-55
    - arguments, 9-56, 9-57
  - `ip6addrctl` command, 1-10, 9-60
  - `ip.cp.pm`, 1-8
    - components, 1-8
  - `ipDefaultTTL` variable, 8-31
  - `ipForwarding` variable, 8-30
  - IPsec support. *See also* strongSwan for OpenVOS, G-1
  - IPv4
    - addresses, 1-3
      - `access_info_monitor` command, 9-7
      - `ifconfig` command, 9-41
      - `ping` command, 9-131, 9-142
      - `telnet` command, 9-160
      - `traceroute` command, 9-175
    - ICMP echo packet, 1-11
    - MTU, 1-3

**IPv6**

- addresses, 1-3
  - access\_info\_monitor command, 9-7
  - ifconfig command, 9-41
  - ping command, 9-131, 9-142
  - telnet command, 9-160
  - traceroute command, 9-175
- ICMPv6 Node Information Node Addresses query, 1-11
- MTU, 1-3
- tftpd6 command, 8-44
- ipv6\_devs variable in the start\_stcp.cm command macro, 2-9
- IP. See Internet Protocol (IP), A-7
- IXGBE driver, 1-8, 1-16, 3-15
  - verifying configuration, 3-25, 6-14

**K**

- Keepalive option, TELNET, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
- kernel\_loadable\_library directory
  - igb.cp.pm, 3-25
  - ixgbe.cp.pm, 3-25
  - sdlmux.cp.pm, 1-16
  - verifying that SDLMUX is loaded, 3-25
- Kernel-loadable drivers, 1-5, 1-8
- Keyword and value fields
  - nsswitch.conf database file, 5-29
  - resolv.conf database file, 5-43
- kill command
  - inetd daemon process, 8-20
  - inetd.conf database file, 5-22

**L**

- LAND attack, 7-2
- Library paths
  - establishing, 6-3
  - verifying, 6-13
- Line speed of PCI adapters, 3-17
- Linger option, TELNET, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
- Link state advertisement (LSA)
  - described, A-19
  - transmittal delay, 5-34
- Link state database (LSD), 8-24, A-19
  - displaying, 9-111
  - synchronizing, A-19

- list\_area\_information subcommand, 9-111
- list\_database subcommand, 9-111
- list\_default\_library\_paths command, 6-13
- list\_destinations subcommand, 9-109
- list\_devices command, 6-17
  - Ethernet PCI adapters, 3-21
  - STREAMS devices, 3-20
- list\_errors subcommand, 9-111
- list\_general\_information subcommand, 9-111
- list\_history subcommand, 9-109
- list\_interfaces subcommand, 9-111
- list\_kernel\_programs command, 3-25, 6-13
- list\_library\_paths command, 6-13
- list\_local\_commands subcommand, 9-109
- list\_lsa subcommand, 9-111, 9-112
- list\_neighbors subcommand, 9-112
- list\_packet\_types subcommand, 9-112
- list\_queues subcommand, 9-112
- list\_ranges subcommand, 9-112
- list\_remote\_commands subcommand, 9-109
- list\_stcp\_params request of
  - analyze\_system, 6-21, 7-1, 7-4
- list\_tree subcommand, 9-113
- list\_users command, 6-12, 6-16
  - verifying daemon processes, 8-2
- Loading drivers
  - IGB, 3-25
  - IXGBE, 3-25
  - SDLMUX
    - automatically, 3-22, 3-23, 6-4, 9-17
    - manually, 3-24
- Local hosts, 5-8
- Log files, 1-6, 6-8, 8-2
  - bootplog, 8-5
  - discarded packets, 7-3
  - ftpd.out, 8-7, 8-8
  - inetd.out, 8-19, 8-20
  - MST server, F-13
  - mst\_server\_name.out, F-10
  - ospfd.io, 8-23
  - ospfd.log, 8-23
  - security attacks, 7-3
  - snmpd.out, 8-30

`syserr_log.date`, 7-2, 7-3, 8-8, 8-12,  
     9-18, 10-5, 10-6  
`tcpdallow`, 5-11, 5-12, 8-10, 8-12, 8-13,  
     8-17, 8-33, 8-35, 8-36, 8-37, 8-38  
`tcpddeny`, 5-11, 5-12, 8-10, 8-12, 8-13,  
     8-17, 8-33, 8-35, 8-36, 8-37, 8-38  
`telnetd.out`, 8-38  
`tftpd.out`, 8-40  
`tftplug`, 8-40, 8-44  
`log_to_file` subcommand, 9-109  
`log_to_stdio` subcommand, 9-109  
 Logical network interface, 3-6  
     defining, A-4  
     logical configuration, 3-6  
     physical configuration, 3-6  
     software considerations, 3-10  
 Login devices, 4-8, 4-12  
     virtual terminal (vterm), F-5, F-11  
 Login service, TELNET, 5-54  
`login_slave` field, `devices.tin` file, 4-15,  
     4-18, 4-24, 4-25, 4-28  
`logs` directory, 6-7  
 Loopback driver  
     device driver configuration, 4-6  
     program file, 1-9  
 Loopback interfaces, 5-8  
 Loose source routing, 9-134  
 Lower-level device drivers of SDLMUX, 1-16  
     IGB, 1-8, 1-16, 3-15  
     IXGBE, 1-8, 1-16, 3-15  
 LSD. *See* Link state database (LSD)

## M

MAC address, 9-11  
 Machine-executable files, 3-2, 4-30  
 Manual hardware failover, 9-16, 9-19  
 Masks, subnet  
     specifying, 9-142, A-13  
     variable-length, A-15  
 Maximum number of sockets, 1-7  
 Maximum transmission unit (MTU), 1-3, 9-57  
     controlling with the `ip_pair_admin`  
         command, 9-55  
     `ifconfig` command argument, 9-42  
     `ping` command argument, 9-132  
     `route` command argument, 9-142  
`mddmon_start_up.cm` command macro, 6-7

MIB files, 5-49  
     OpenVOS, 9-69, 9-93  
     SNMP, 5-49, 9-54  
`minor_number` field, `devices.tin` file, 4-10,  
     4-12  
`module_name` field, `devices.tin` file, 4-3,  
     4-4, 4-5, 4-6, 4-7, 4-10, 4-11, 4-18,  
     4-25, 4-28  
`module_start_up.cm` file, 6-3  
     initializing partnered Ethernet PCI  
         adapters, 3-10  
     loading SDLMUX, 3-10  
 Modules  
     configuring STCP, 6-2  
     STCP requirements, 1-17  
 MST server. *See* Multisession TELNET (MST)  
     server  
 MTU. *See* Maximum transmission unit (MTU)  
 Multicasting, A-7  
     specifying an alias, 9-145  
     specifying default interface, 9-145  
 Multihoming, 9-48  
 Multisession TELNET (MST) server, 1-2, F-1

## N

`name` field, `devices.tin` file, 4-3, 4-4, 4-5,  
     4-6, 4-7, 4-10, 4-11, 4-15, 4-17, 4-24,  
     4-25, 4-27  
 NBMA interfaces, 5-34  
`netstat` command, 1-10, 9-61  
     AF UNIX protocol statistics, 9-93  
     arguments, 9-62, 9-63, 9-65  
     ARP protocol statistics, 9-74  
     ICMP protocol statistics, 9-75  
     UDP protocol statistics, 9-92  
     verifying daemon processes, 8-2  
 Network interface cards (NICs). *See* Network  
     interfaces  
 Network interfaces  
     adding, 6-21  
     adding an IP address to, 6-21, 6-22, 9-43  
     assigning an IP address, 9-39  
     changing state of, 6-21, 9-39  
     changing status of, 9-37  
     checking status, 9-16  
     configuring and verifying, 9-37

- configuring LANs
  - multiple, A-10
  - single, A-12
- creating `devices.tin` file entries, 3-12
- definition, 1-15
- deleting, 6-21, 9-41
- Ethernet PCI adapters, 1-15, 1-16
- logical, defining, A-4
- name, 3-14
- OSPF configuration, 5-34
- partnered, 1-15, 3-3, 3-6, 3-16, 3-18
  - definition, 1-15
- physical, 1-15
- SDLMUX, 1-15
- simplex, 1-15
- software configuration, 1-17
- specifying, 9-39
- status, 9-61
- verifying, 6-18, 9-43

`networks` database file, 1-13, 9-141, B-9

- described, 5-26
- editing, 5-26
- fields, 5-26
- identifying subnets, A-4

`ngrep` command, 1-10, 9-106

NICs. *See* Network interfaces

`nobody.nobody` user, 1-9, 1-12, 8-41, 9-171

No-delay option, TELNET, 5-54, 5-55, 8-37, 9-165, 9-167, A-21

`nsswitch.conf` database file, 1-13

- described, 5-28
- editing, 5-28
- fields, 5-29

## O

Object library, 1-6

`omon` command, 9-107

- configuring a session, 9-109
- examples, 9-114–9-115
- listing commands, 9-109
- `omonconf` file, 9-108
- output logging, 9-109
- subcommands, 9-109

`omon` daemon process, 1-11

`omonconf` database file, 1-13, 5-3, 9-107, 9-108

Open Shortest Path First (OSPF). *See* OSPF *and* `ospfd` daemon process

Options, STCP, A-21

Options, TELNET

- keepalive, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
- linger, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
- no-delay, 5-54, 5-55, 8-37, 9-165, 9-167, A-21

OSPF, 8-23

- area information, displaying, 9-111
- backbone area, A-20
- host routes, listing, 9-111
- neighbors, 9-112
- routers
  - destination file defined, 9-107
  - electing a designated router, 5-35
  - monitoring, 9-107
  - specifying in an `omon` subcommand, 9-109
- routing table, 9-112
- timer queue, 9-112

`ospfd` daemon process, 1-11, 8-23

- configuration checklist, 2-7, 2-9
- debug logging, 10-6
- event logging, 10-6
- logging, 9-113
- packet logging, 10-6
- `start_stcp.cm` command macro, 8-24, 8-25
- starting, 8-25
- stopping, 9-113
- troubleshooting, 10-6

`ospfdconf` database file, 1-13, B-10

- fields, 5-32
- sections, 5-32
- template, 2-7, 5-31, 5-37
- updating, 5-31

`ospfd.io` file, 9-109

- described, 10-6
- error messages, E-1
- log file, 8-23

`ospfd.log` file, 10-6

- log file, 8-23

`ospfd.pkt` file, 10-6

Outgoing slave devices, 4-8, 4-26

- device configuration, 4-26

## P

Packet logging, OSPF, 9-110, 10-6  
 Packet types log, displaying, 9-112  
 packet\_monitor command, 1-11, 9-117  
   arguments, 9-118, 9-120, 9-121, 9-122, 9-123, 9-126  
   examples, 9-125  
   filters, 9-124, 9-125  
 Packets  
   ARP response, 9-12  
   broadcast, forwarding, 9-42  
   probe, 9-177  
 parameters field of devices.tin file, 3-16, 4-18, 4-28  
   -duplex, 3-17  
   -input\_buffers, 3-17  
   -min\_speed, 3-17  
   -mtu, 3-17  
   -no\_energy\_efficient\_ethernet, 3-17  
   -output\_buffers, 3-18  
   -partner, 3-10, 3-16, 3-18  
   -rings, 3-17  
   -sdlmux, 3-10, 3-16  
   -speed, 3-17  
   -use\_minimum\_vm, 3-18  
 Paranoid checks of TCP wrappers  
   functionality, 5-12  
 Partnered network interfaces, 1-15  
   adding a partner, 3-29, 9-16  
   causing a hardware failover, 9-16  
   configuration requirements, 3-3, 3-6  
   defining two new interfaces as partners, 3-25  
   definition, 1-15  
   deleting a partner, 3-28, 3-29, 9-16, 9-19  
   devices.tin entries, 3-16, 3-18  
   Ethernet PCI adapters, 1-16  
   ifconfig command, 3-25  
   initializing an SDLMUX-group interface, 3-21, 9-17  
   requirements for configuring, 3-6  
   SDLMUX, 1-15, 3-16, 3-18  
     software considerations, 3-10  
     status, 9-17, 9-18  
     uninitializing a partner, 9-17  
   SDLMUX-group interfaces, 1-15

## Password

  ftpd daemon process, 8-13, 8-42  
   OSPFD daemon process, 5-35  
   tftpd daemon process, 8-42  
 path MTU discovery, 1-3  
   controlling with the ip\_pair\_admin command, 9-55  
 Physical configuration of Ethernet PCI adapters and SDLMUX, 3-3  
 Physical network interfaces, 1-15  
 ping command, 1-11, 6-18, 9-129, 9-178  
   arguments, 9-130, 9-134, 9-135  
   examples, 9-135  
 Pipe devices, TELNET, 4-8  
 pkt\_logging subcommand, 9-113  
 Planning the SDLMUX configuration, 3-3  
   logical configuration, 3-6  
   physical configuration, 3-3  
 Point-to-point routes, in ospfdconf file, 5-32  
 Port scanning, 7-4  
 Ports  
   Ethernet PCI adapters, 1-15, 1-16  
   FTP, 8-11  
   ftp command argument, 9-26  
   local services, 5-47  
   TELNET, 4-19, 4-28, 9-159  
     incoming slave devices, C-5  
     outgoing slave devices, C-8  
 POSIX.1 compliance, 1-2  
 Priority, OSPF router, 5-35  
 priv\_terminal field, devices.tin file, 4-16, 4-18, 4-24  
 Privileged terminal setting, 4-13, 4-18, 4-25, 5-56  
 Probe packets, 9-177  
 Problems, diagnosing, 10-1  
 Procedures, starting STCP, 6-11  
 Processes  
   *See also* Daemon processes  
   bootpd daemon process, 1-9  
   ftpd daemon process, 1-10  
   inetd daemon process, 1-10  
   listing, 8-3  
   MST server, F-8  
   omon daemon process, 1-11  
   ospfd daemon process, 1-11  
   rtsold daemon process, 1-11  
   snmpd daemon process, 1-11  
   starting and stopping, 1-5

- STCP networking daemon processes, 8-1
- `sync_cfgd` daemon process, 1-11
- `tcpd` daemon process, 1-11
- `telnet_msd` daemon process, 1-12
- `telnetd` daemon process, 1-12
- `tftpd` daemon process, 1-12
- `tftpd6` daemon process, 1-12
- `xinetd` daemon process, 1-12
- Protocol drivers
  - configuring
    - AF UNIX, 4-7
    - IP, 4-5
    - loopback, 4-6
    - STCP, 4-2
    - UDP, 4-4
  - confirming configuration of, 6-17
- Protocol statistics
  - AF UNIX, 9-93
  - UDP, 9-92
- Protocols
  - See *also* individual protocol names
  - adding to `protocols` database file, 5-40
  - Internet Protocol (IP), A-3
- `protocols` database file
  - described, 5-40
  - editing, 5-41
  - fields, 5-41
- Pseudo-terminals, 4-2

## R

- RANGE entry in `ospfdconf` file, 5-33
- Rebooting, 6-11
- Request for Comments (RFCs), A-4
  - RFC 1123, 5-9, 5-27, 5-41, 5-47
  - RFC 1878, A-15
  - RFC 952, 5-27, 5-41, 5-47
  - RFC 959, A-4
- `resolv.conf` database file, 1-14, A-5
  - described, 5-42
  - editing, 5-43
  - fields, 5-43
  - name servers, specifying, 5-42
- Retransmit interval, 5-34
- RFCs. See Request for Comments (RFCs)
- Root, A-23
- `root.root` user, 1-6, 1-10, 1-14, 5-23, 5-24, 5-57, 6-1, 8-19

- `route` command, 1-11, 9-140
  - arguments, 9-141, 9-142
  - examples, 9-146
  - specifying default system interface, 9-145
  - `start_stcp.cm` command macro, 6-8
  - VLSN configuration, A-17
- Router-dead interval, 5-35
- Routers/Gateways
  - adding to `hosts` database file, 5-9
  - defined, A-1
  - enabling modules as
    - module as, 1-18
  - OSPF
    - monitoring, 9-107
    - specifying in an `omon` subcommand, 9-109
- Routes, 9-144
  - configuring, 6-9, 9-140
  - deleting, 9-141
  - OSPF, adding and deleting, 9-110
  - overview of, A-18
  - routing tables, 9-64
  - specifying, 6-9, 9-140
  - subnetwork, A-18
  - tracing, 9-174
- Routing information, 9-107
- Routing table, verifying entries in, 6-19
- RTRID entry in `ospfdconf` file, 5-32
- `rtsol` command, 9-154
- `rtsold` daemon process, 1-11, 8-26

## S

- SAMBA file server and the `swat` daemon, 5-24
- Sample device entries of SDLMUX-group
  - interfaces, B-5
- `sample_start_inetd.release_number` file, 1-14
- `sample_start_stcp.release_number` file, 1-14
- `sample_start_xinetd.release_number` file, 1-14
- `sample_stop_inetd.release_number` file, 1-14
- `sample_stop_xinetd.release_number` file, 1-14
- SDLMUX, 1-9
  - configuration procedures, 3-3
  - devices, 3-12



- initializing, 9-17
- status, 9-17, 9-18
- uninitializing, 9-17
- documentation, 1-16
- dump\_sdlnmux request of
  - analyze\_system, 9-18
- Ethernet PCI adapters, 1-15
- group. *See* SDLMUX-group interfaces
- loading, 3-23
  - automatically, 3-22, 3-23, 6-4, 9-17
  - manually, 3-24
- lower-level device drivers, 1-8, 1-15, 1-16
- network interfaces, 1-15
- overview, 1-16
- planning the configuration, 3-3
- software configuration procedures, 3-10
- tasks, 1-16
- upper-level protocol stack, 1-9
- verifying, 3-25, 6-14
  - devices, 3-20
- sdlnmux\_status action of the dlnmux\_admin command, 9-17, 9-18
- sdlnmux.cp.pm program module, 1-16
- SDLMUX-group interfaces
  - configuring, 3-11, 3-12
  - definition, 1-15
  - initializing, 3-21, 9-17
  - name, 3-13
  - partnered Ethernet PCI adapters, 1-15
  - planning, 3-6
  - sample device entries, 3-13
  - simplexeth Ethernet PCI adapters, 1-15
  - software considerations, 3-10
  - status, 9-17, 9-18
- Security, 7-1
  - DOS attack, 7-2
  - FTP and ftpd.out, 8-8
  - LAND attack, 7-2
  - logging attacks, 7-3
  - port scanning, 7-4
  - STCP driver, 4-30
  - STCP parameters, 7-1
- security\_check\_file, ftpd daemon process argument, 8-16
- Servers
  - See also* Daemon processes
  - name, and OpenVOS, 5-43
- services database file, 1-14, 8-11, 9-161, 9-164
  - described, 5-46
  - editing, 5-47
  - fields, 5-47, 5-57
  - ftpd daemon process, 8-11
  - inetd.conf database file, 5-21
  - sample TELNET entries, 8-38
  - telnetd daemon process, 9-169
  - telnet-service database file, C-3
- Services, TELNET, 5-53
  - login, 4-12
  - slave, 4-20, 4-26, C-4
- set\_stcp\_param request of
  - analyze\_system, 6-21, 7-1
- set\_udp\_ordering command, 9-155
  - arguments, 9-154, 9-155
- setkey command, 9-157
- Setting the line speed of an adapter, 3-17
- Simple Network Management Protocol (SNMP)
  - client software, 5-50
  - communities, 5-52
  - daemon (snmpd) command, 8-29
  - manager, obtaining, 1-18
  - STCP variables, 5-50, 8-29, 8-30
- Simplexeth network interfaces, 1-15
  - definition, 1-15
  - Ethernet PCI adapters, 1-15
    - adding, 3-30
    - configuration requirements, 3-3, 3-6, 3-25
  - SDLMUX, 1-15
- Slave devices
  - incoming, 4-20, C-1, C-3
  - outgoing, 4-26, C-5
  - virtual terminal (vterm), F-5, F-11
- SNMP. *See* Simple Network Management Protocol (SNMP)
- snmpconf database file, 1-14, 8-29
  - described, 5-49
  - editing, 5-50
  - system variables, 5-50
- snmpd daemon process, 1-11, 5-50
  - community setting, 5-52
  - services database file, 8-30
  - snmpconf database file, 8-29
  - snmpd.out log file, 8-30
  - start\_stcp.cm command macro, 8-30
  - snmpd.out log file, 8-30

- Software configuration procedures of
  - SDLMUX, 3-10
- Software requirements of STCP, 1-17
- `sort_database` subcommand, 9-113
- Source routing
  - loose, 9-134
  - strict, 9-135
- Specifying partnered network interfaces, 9-16
  - SDLMUX, 3-10
- Speed setting of PCI adapters, 3-17
- `start_inetd.cm` command macro, 1-6, 1-14, 2-4, 6-1, 6-10, 6-11, 8-20
- `start_process` command
  - `ftpd` daemon process, 8-11
  - `inetd` daemon process, 8-20
  - `snmpd` daemon process, 8-31
  - `telnetd` daemon process, 8-36
- `start_rsn` command, 6-7
- `start_stcp_stack` internal command, 1-11
- `start_stcp.cm` command macro, 2-3, 6-4
  - `ipv6_devs` variable, 2-9
  - `ospfd` entries, 2-7, 8-25
  - SDLMUX, 3-10, 3-11, 3-25, 3-27, 3-30
- `start_xinetd.cm` command macro, 1-14, 6-10
- States
  - network interface, 9-40
  - TCP protocol, 9-71
- Static routes, OSPF, adding and deleting, 9-110
- Statistics
  - AF UNIX, 9-93
  - errors indicated by, 9-95
  - UDP, 9-92
- Status
  - network interfaces, 9-37, 9-61
  - partnered network interfaces, 9-16, 9-17, 9-18
  - TELNET services, 9-168, 9-169
- STCP. *See* STREAMS TCP/IP (STCP)
- `stcp_inetd`, `inetd` process name, 8-20
- `stcp_meters` request of
  - `analyze_system`, 6-21
- `stop_inetd.cm` command macro, 1-6, 1-14, 6-2, 8-21
- `stop_xinetd.cm` command macro, 1-14
- STREAMS Data-Link Multiplexer (SDLMUX)
  - defined, 1-15
- `streams` device type, 3-13
- STREAMS devices, 4-1
  - listing, 6-17
- STREAMS drivers
  - setting access to, 4-30
  - STCP, 4-2
  - `tli_term`, 4-8
- STREAMS modules, setting access to, 4-30
- STREAMS TCP/IP (STCP), 1-1
  - adding a partner, 3-29
  - application programming interface, 1-6
  - command library, 1-6
  - command library paths, 6-3
  - components, 1-5
  - components illustration, 1-7
  - configuration checklist, 2-1
  - database file templates, 1-6
  - defining new interfaces as partners, 3-6
  - device configuration, confirming, 6-17
  - device entries for network interfaces, 3-16, 3-18
  - device types, 4-1
  - devices, 4-1
  - directories, 1-5
  - drivers
    - confirming, 6-13
    - controlling access to, 4-30
    - described, 1-9
    - device entry, 4-2
  - initialization, confirming, 6-13
  - kernel-loadable components, 1-5
  - log files, 1-6
  - overview of, 1-1
  - protocol drivers
    - IP, 4-5
    - loopback, 4-6
    - UDP, 4-4
  - routes, 9-144
  - security, 7-1
  - simplex Ethernet PCI adapters, 1-15
  - software requirements, 1-17
  - specifying routes, 6-9
  - starting, 2-1, 6-11
  - troubleshooting, 10-1
- `Streams_Daemon` daemon process, 6-2
  - verifying, 6-12
- `streams_driver` field, `devices.tin` file, 4-3, 4-5, 4-6, 4-7, 4-8, 4-10, 4-11

- Streams\_Memory\_Daemon daemon
  - process, 6-2
  - verifying, 6-12
- streams\_pci device type, 3-15
- Strict source routing, 9-135
- strongSwan for OpenVOS, G-1
  - avoiding problems when tunneling, G-9
  - configuration files, G-4
  - default security policy, G-6
  - documentation, G-2
  - important considerations, G-10
  - installing, G-3
  - man pages, G-4
  - software requirements, G-2
  - starting and stopping the daemon, G-4
  - supported algorithms, G-5
- Stub area
  - configuring, 5-33
  - defined, A-21
- Subcommands, omon command, 9-109
- Subnet masks, A-13
  - listed in RFC 1878, A-15
  - setting, 9-40
  - specifying, 9-40, 9-142, A-13
  - specifying with the route command, 6-9, 9-142
  - variable length, A-15
  - VLSN configuration, A-17
- Subnetworks
  - configuring, A-14
  - defined, A-1
  - establishing communication routes, A-18
  - identifying in networks database file, A-4
  - specifying network masks, A-13
- swat daemon, 5-24
- sync\_cfgd daemon process, 1-11, 5-3
- sync\_cfgd.conf file, 1-15, 5-3
- sync\_cfgd.leases file, 1-15
- syserr\_log.date file
  - discarded connect requests, 7-3
  - FTP or TELNET connections, 10-5
  - ftpd daemon, 8-8, 8-12
  - OSPF route entry, 10-6
  - packet attacks, 7-2, 7-3
  - route -flush entry, 10-6
  - SDLMUX messages, 9-18
- >system>acl directory, 4-30

## T

- TCP. See Transmission Control Protocol (TCP)
- TCP wrappers functionality
  - ftpd daemon process, 8-10
  - tcpd daemon process, 8-32
  - telnetd daemon process, 8-35
- tcpd daemon process, 1-11, 8-32
  - bootpd daemon process, 8-5
  - inetd.conf database file, 5-21, 5-23
- tcpdallow log file, 5-11, 5-12, 8-10, 8-12, 8-13, 8-17, 8-33, 8-35, 8-36, 8-37, 8-38
- tcpddeny log file, 5-11, 5-12, 8-10, 8-12, 8-13, 8-17, 8-33, 8-35, 8-36, 8-37, 8-38
- tcpdump command, 9-158
- TELNET, 1-2
  - confirming configuration of, 6-18
  - connections
    - client-initiated, 4-20, 4-26, 4-28
    - host-initiated, 4-26, 4-28, 4-29, C-1, C-5, C-7
  - daemon process, 1-12, 4-10, 8-34
  - defining incoming slave devices, 4-20
  - defining login devices, 4-12
  - defining pipe entries, 4-11
  - defining the daemon process, 6-8
  - defining TLI access-layer device
    - driver, 4-9
  - devices, 4-8
  - drivers, 1-9
  - host-initiated connections, 4-28
  - login devices, configuring, 4-12
  - login service, 5-54
  - MST server, F-1
  - options
    - keepalive, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
    - linger, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
    - no-delay, 5-54, 5-55, 8-37, 9-165, 9-167, A-21
  - pipe device entries, 4-11
  - services
    - deleting, 9-164
    - verifying status, 9-168, 9-169
  - services, defining, 5-54, 9-163
  - slave devices, 4-24
    - applications attaching to, C-1
    - confirming configuration of, 6-18

- incoming, 4-26, 4-28
- incoming slaves, C-1, C-7
- outgoing, 4-26, 4-28, 4-29
- outgoing slaves, 4-28, C-5
- TCP wrappers functionality, 8-37, 9-161
- telnet\_msd daemon process, 1-2, 1-12, F-8
- telnetd daemon process, 1-2, 8-34
- tli\_al access-layer device driver, 4-10
- virtual terminal (vterm) devices, F-2
  - login, F-5, F-11
  - slave, F-5, F-11
- window terminal devices, 4-18, 6-18
- telnet command, 1-12, 9-2, 9-159
  - arguments, 9-159
  - examples, 9-161
- telnet\_admin command, 1-12, 6-23, 9-163, C-4
  - arguments, 9-164, 9-165, 9-166
  - device configuration, 4-8
  - examples, 9-168
- telnet\_msd daemon process, 1-2, F-8
- telnet\_msd.pm, 1-12, F-1
- telnetd daemon process, 1-2, 8-34, C-5
  - configuration checklist, 2-5
  - device configuration, 4-8
  - services database file, 8-36
  - slave devices, 4-20, 4-26, C-4
  - start\_stcp.cm command macro, 8-36
  - TCP wrappers functionality, 8-35
  - telnet\_admin command, 8-36
  - telnetd.out log file, 8-38
  - telnetSERVICE database file, 5-53, 8-35
- telnetd.out log file, 8-38
- telnetSERVICE database file, 1-15, 9-163, 9-164, 9-165, 9-166
  - described, 5-53
  - editing, 5-55
  - fields, 5-55, 5-56
  - sample entries, 8-38
  - services database file, 9-169, C-3
  - telnetd daemon process, 9-169, C-4
- Template files, STCP
  - database files, 1-6, 5-2
  - start\_inetd.cm command macro, 6-10
  - start\_stcp.cm command macro, 6-7
  - start\_xinetd.cm command macro, 6-10
- terminal\_type field, devices.tin file, 4-18, 4-25, 4-28
- tftp command, 9-170
  - arguments, 9-170
  - examples, 9-172
- TFTP. See Trivial File Transfer Protocol (TFTP)
- tftp6 command, 9-173
- tftpd daemon process, 8-40
  - inetd.conf database file, 8-41
  - start\_inetd.cm command macro, 8-41
  - tftpd.out log file, 8-40
- tftpd6 daemon process, 8-44
- tftpd6.pm daemon process, 1-12
- tftpd.out log file, 8-40
- tftpd.pm program module, 1-12
- tftplog log file, 8-40, 8-44
- time service, 5-22
- TLI. See Transport Layer Interface (TLI)
- tli\_al access-layer device driver, 4-8, 4-10
- tnmod.cp.pm TELNET driver, 1-8
- tpipe.cp.pm TELNET pipe driver, 1-8
- traceroute command, 9-174
  - arguments, 9-175
  - examples, 9-178
- Transit area, OSPF
  - configuring, 5-33
- Transmission Control Protocol (TCP), 1-7
  - TCP wrappers functionality, 1-1
    - bootpd daemon process, 8-5
    - configuring, 2-7
    - hosts.allow database files, 5-10
    - hosts.deny database files, 5-10
    - inetd daemon process, 5-21
    - inetd.conf database file, 5-21
    - paranoid checks, 5-12
    - tcpd daemon process, 8-32
    - telnet command, 9-161
    - tftp command, 9-171
    - unsupported features, 5-10
    - verifying connectivity, 6-20
- Transmission services
  - IP multicast, A-7
  - unicast, A-7
- Transmit delay, 5-34
- Transport Layer Interface (TLI)
- Trivial File Transfer Protocol (TFTP), 9-170
  - TCP wrappers functionality, 9-171
- Troubleshooting STCP, 10-1
  - basic checks, 10-1, 10-2

- incoming TELNET or FTP
  - connections, 10-5
- OSPF, 10-6
- outgoing TELNET or FTP
  - connections, 10-5
- reaching hosts on different subnets, 10-4
- reaching hosts on the local subnet, 10-2
- reaching one host on the local
  - subnet, 10-4
- using the `arp -all` command, 10-4
- using the `ifconfig -all`
  - command, 10-2
- using the `list_kernel_programs`
  - command, 10-1
- using the `netstat` command, 10-1
- using the `packet_monitor`
  - command, 10-3
- using the `ping` command, 10-2, 10-3, 10-4
- using the `route print` command, 10-2

## U

- UDP driver, 1-9
- UDP. *See* User Datagram Protocol (UDP)
- Unicast transmission service, A-7
- `uninit_sdlmux` action of the `dlnux_admin`
  - command, 9-17
- Uninitializing
  - Ethernet PCI adapters, 9-17
  - SDLMUX groups, 9-17
  - SDLMUX-group interfaces, 9-17
- Upper-level protocol stacks and SDLMUX, 1-9
- User Datagram Protocol (UDP), 1-9
  - driver device configuration, 4-4
- Users
  - `nobody.nobody`, 1-9, 1-12, 8-4, 8-41, 9-171
  - `root.root`, 1-6, 1-10, 1-14, 5-23, 5-24, 5-57, 6-1, 8-19

## V

- Variable-length subnets (VLSN)
  - creating, A-14
  - sample configuration, A-16, A-17
  - STCP support, 1-2
- Verifying configurations
  - devices, 3-20
  - IGB, 3-25, 6-14

- IXGBE, 3-25, 6-14
- SDLMUX, 3-25, 6-14
- VIRTUAL entry in `ospfdconf` file, 5-36
- Virtual link
  - defined, A-20
  - illustration, A-20
- Virtual terminal (vterm) devices, F-2
  - configuring, F-2
  - login, F-5, F-11
  - slave, F-5, F-11
  - TELNET, F-2
- VLSN. *See* Variable-length subnets (VLSN)

## W

- Window terminal devices, 5-53
  - applications attaching to, C-1
  - confirming configuration of, 6-18
  - login devices, TELNET, 4-18
  - slave devices, TELNET
    - client-initiated connections, C-1
    - host-initiated connections, C-5

## X

- `xinetd` daemon process, 1-12, 8-45
- `xinetd.conf` database file, 1-15

