



SunScreen 3.2 Administrator's Overview

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 806-6347
September 2001

Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road Palo Alto, CA 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2001 Sun Microsystems, Inc. 901 San Antonio Road Palo Alto, CA 94303-4900 U.S.A. Tous droits réservés

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPOUDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



010910@2471



Contents

Preface	15
1 SunScreen Overview	21
What Is SunScreen?	21
Software and Hardware Requirements	23
Required Patches	25
Java Plug-In Software	25
Compatibility With Other SunScreen Products	25
SunScreen Lite	26
Supported Features	26
Limitations	27
Online Help and Documentation	27
2 SunScreen Concepts	29
Why Use SunScreen?	29
A Sample SunScreen Network Map	29
How SunScreen Works	30
Routing and Stealth Mode Interfaces	32
Routing Mode Interface	32
Stealth Mode Interface	33
Administration	33
Local Administration	34
Remote Administration	34
Locating the SunScreen Screen	35
Security Policy	36

	Configuration	37
	Stateful Packet Filtering	37
	Centralized Management Group	37
	Network Address Translation (NAT)	37
	Tunneling and Virtual Private Networks (VPN)	38
	High Availability (HA)	39
	Encryption	39
	Logging	39
	Proxies	40
	Event Logging With Proxies	40
	IPsec/IKE	41
	Internet Key Exchange (IKE)	42
	SunScreen IPsec Configuration	43
3	Packet Filtering	45
	SunScreen Model	45
	Stateful Packet Filtering	46
	Inbound Packet Rule Checking	47
	Outbound Packet Rule Checking	47
	Policy Rules	48
	Sequence of Rules	48
	Services and Service Groups in Rules	49
	Rule Syntax	50
	Example of a Rule Configuration	51
	Policy Versions	52
4	Common Objects	55
	Service Object	56
	Single Service	57
	Service Group	59
	Address Object	60
	Host Address	61
	Address Range	61
	Address Group	62
	Designing an Addressing Scheme	63
	Screen Object	64

	Miscellaneous Parameters	65
	SNMP Information	66
	HA and CMG Parameters	67
	Mail Proxy Configuration	68
	Interface Object	68
	Routing Interface	71
	Stealth Interface	71
	Admin Interface	72
	Routing and Stealth Interfaces on a Single Screen	72
	HA Interface	72
	Disabled Interface	72
	authuser Object	73
	Time Object	75
	proxyuser Object	77
	Jar Signature Object	78
	Jar Hash Object	79
	Certificate Object	79
	Single Certificate	80
	Certificate Group	80
	IPsec Key Object	81
	Administration GUI Limitations	81
5	Administration	83
	Administering the Screen	83
	Local Administration	84
	Remote Administration	84
	Centralized Management Group	85
	Logging	86
	Common Objects Used in Centralized Administration	86
	Creating Common Objects and Policies for Multiple Screens	86
6	Encryption, Tunneling, and Virtual Private Networks	89
	Encryption and Decryption	89
	How SunScreen Uses Encryption	90
	Packet Examination	90
	Tunneling	91

	Using IKE With SunScreen	93
	IKE Options	94
	IKE Policy Rules	96
	Defining a VPN	98
	Adding a VPN Rule	101
	VPN Limitations	103
7	Network Address Translation	105
	Network Address Translation	105
	NAT Rules	107
	Static NAT	107
	One-to-One Translations	108
	Address Range to Another Address Range	108
	Dynamic NAT	109
	Dynamic NAT Collisions	110
	Choosing NAT Addresses	112
	NAT Examples	112
	Example One	113
	Example Two	113
	Routing Interface Examples	116
	Stealth Interface Examples	116
	Applying NAT	116
8	High Availability	119
	Why High Availability?	119
	Hardware Requirements	119
	SunScreen HA Definitions	120
	SunScreen HA Configurations	121
	Basic SunScreen HA in Routing Mode	121
	Remotely Administered SunScreen HA Configuration in Routing or Stealth Mode	123
	HA Using Switches	125
	HA Network Connections and Failovers	125
	Configuring HA	126
	Administering HA	127

9	Authentication	129
	User Authentication	129
	User Identification	129
	Authorized User	130
	Defining an Authorized User Object	130
	Creating an Authorized User Object	132
	Authorized User Authentication Processing Logic	134
	Administrative User	135
	Proxy Users	135
	Defining a Proxy User Object	137
	Creating a Proxy User Object	138
	Proxy User Processing Logic	141
	RADIUS User Authentication Details	143
	RADIUS Server Configuration	144
	RADIUS Node Secret Configuration	145
	Typical RADIUS Configuration	146
	Other vars for RADIUS Configuration	147
	Other RADIUS Protocol Notes	148
	RADIUS Testing	148
	RADIUS Usage	148
	SecurID User Authentication Processing Details	149
	ACE/Client, ACE/Agent, and the SunScreen Stub Client	150
	SecurID Access Paths	152
	SecurID PIN Establishment	153
	Typical SecurID Configuration	154
	Other SecurID Details	157
	SecurID Usage	157
10	Proxies	159
	SunScreen Proxies	159
	How Proxies Work	160
	Policy Rule Matching	161
	Proxy User Authentication	162
	Proxy Limitations	162
	Automatically-Saved Common Objects	163
	FTP Proxy	163
	FTP Proxy Operation	163

FTP Proxy and Anonymous FTP	164
FTP Proxy Use	165
Other FTP Proxy Issues	165
HTTP Proxy	166
HTTP Proxy Port Restrictions	167
HTTP Proxy Access for ftp://	168
HTTP Proxy User Authentication	170
HTTP Proxy Operation	171
SMTP Proxy	173
SMTP Proxy Operation	174
Spam Control	175
Examples	175
Relay Control	176
Other Mail Configuration Issues	178
SMTP Proxy Rules	179
Telnet Proxy	179
Telnet Proxy Operation	179
Other Telnet Proxy Issues	180
Telnet Proxy Use	180
Using Encryption With Proxies	181
VirusWall Content Scanning	181
HTTP Proxy Access to VirusWall	181
SMTP Proxy Access to VirusWall	184
VirusWall Setup Issues	186

11 Logging 189

Packet Logging	189
Logging Limitations	190
Log File Locations	190
Configuring Traffic Log Size	190
Configuring the Global Default Log Size	191
Configuring the Log Size for a Specific Screen	192
Configuring Events to be Logged	192
Size of Logged Items	194
Level of Logging	195
Configuring Log Event Limiters	195
Log Retrieval and Clearing	196

Examples: get, clear, get_and_clear	197
Log Statistics	198
ssadm logstats Subcommand	198
Log Inspection and Browsing	199
Log Filters and the logdump Command	199
logdump Extensions	200
SunScreen welfmt Utility	201
HTTP Proxy Header Logging	202
Logged Network Packet Enhancements	203
General Event Type Enhancements	204
Log Record Format	204
Extended Log Event Enhancements	205
Log Filtering Macros	206
Displaying and Creating Log Macros	207
Log Macro Name and Body	209
Listing Log Macros	210
Log Macro Usage	212
A Migrating From Earlier SunScreen Firewall Products	215
B Configuration Editor Reference	219
What Is the Configuration Editor?	219
Save Not Required for Some Common Objects	220
Solaris (shell) Commands	221
ssadm	221
ssadm Subcommands	223
ssadm Subcommand Summary	224
ssadm activate	225
ssadm active	226
ssadm algorithm	227
ssadm backup	228
ssadm certdb	228
ssadm certlocal	228
ssadm certrldb	229
ssadm configure	229
ssadm debug_level	229

ssadm edit	230
ssadm ha	230
ssadm lock	231
ssadm log	232
ssadm logdump	232
ssadm login	232
ssadm logout	233
ssadm logmacro	234
ssadm logstats	234
ssadm patch	234
ssadm policy	235
ssadm product	236
ssadm restore	236
ssadm spf2efs	236
ssadm sys_info	236
traffic_stats Subcommand	237
Unsupported Commands	237
ssadm lib/nattables	237
ssadm lib/screeninfo	237
ssadm lib/statetables -f	238
ssadm lib/support	238
ss_client	239
ssadm SKIP Commands	239
Configuration Editor	240
Configuration Editor Data Model	240
Configuration Editor Subcommands	242
add	244
add address	245
add screen	245
add service	247
add interface	248
add certificate	249
add key	250
add time	250
add rule	251
add nat	255
add accesslocal	255

add accessremote	256
add vpngateway	257
add_member	258
authuser	258
delete	259
delete_member	259
insert	260
jar_hash	260
jar_sig	261
list	262
list_name	263
load	263
lock	264
lock_status	264
search	264
move	265
replace	266
refer	266
referlist	266
rename	267
renamereference	268
save	268
saveas	268
reload	269
verify	269
mail_relay	269
mail_spam	270
proxyuser	270
vars	271
quit	271
QUIT	271
Network Monitoring and Maintenance	272
Using the ssadm logdump Command	272
Using the ssadm debug_level Command	272
Gathering Information From Your System to Report Support Issues	273

C	Services and State Engines	275
	Standard Services	275
	* Service	282
	ah Service	283
	archie Service	283
	CoolTalk Service	283
	dns Service	283
	esp Service	283
	ftp Service	284
	ICMP Packets	284
	icmp Service	285
	IP Packets	285
	ip Services	286
	ipsec Service	287
	isakmp Service	287
	ipv6 tunnel Service	287
	nfs readonly Service	287
	ntp Service	287
	realaudio Service	287
	rip Service	288
	rpc Service	288
	smtp (Electronic Mail) Service	289
	sqlnet Service	289
	TCP Services	289
	traceroute Service	290
	tsolpeerinfo Service	290
	UDP Services	291
	VDOLive Service	291
	www (World-Wide-Web Access) Service	292
	Network Service Groups	292
	State Engines	295
	Characteristics of State Engines	295
	dns State Engine	296
	ftp State Engine	299
	icmp State Engine	300
	ip State Engine	300
	ipfwd State Engine	300

ipmobile State Engine	300
iptunnel State Engine	301
nis State Engine	301
ntp State Engine	301
ping State Engine	302
pmap_nis State Engine	302
pmap_tcp State Engine	302
pmap_udp State Engine	303
realaudio State Engine	303
rpc_tcp State Engine	303
rpc_udp State Engine	304
rsh State Engine	304
sqlnet State Engine	304
tcp State Engine	305
tcp_keepalive State Engine	305
tcpall State Engine	306
udp State Engine	306
udpall State Engine	307
udp_datagram State Engine	308
udp_stateless State Engine	308
D Error Messages	309
Error Messages From ssadm edit	309
Error Messages From ssadm activate	313
Error Messages From ssadm lock	321
E Logged Packet Reasons	323
Why Codes	323
Glossary	329
Index	341

Preface

SunScreen™ 3.2 software is part of the family of SunScreen products that provide solutions to security, authentication, and privacy requirements for companies to connect securely and conduct business privately over an insecure public internetwork. Earlier SunScreen firewall products include SunScreen EFS, SunScreen SPF-100, and SunScreen SPF-200, their respective Administration Stations, and SunScreen packet filtering software. This SunScreen product integrates the two SunScreen firewall technologies—SunScreen EFS and SunScreen SPF-200—and includes two encryption technologies: SKIP (Simple Key-Management for Internet Protocols) and IPsec/IKE (Internet Protocol Security/Internet Key Exchange).

SunScreen 3.2 Administrator's Overview contains background and reference information needed to properly configure, monitor, and maintain SunScreen 3.2.

Who Should Use This Book

SunScreen 3.2 Administrator's Overview is intended for system administrators responsible for the operation, support, and maintenance of network security. This manual assumes that you are familiar with UNIX® system administration, TCP/IP networking concepts, and your network topology.

Before You Read This Book

You need to be familiar with the following information before you install and administer SunScreen 3.2:

- SunScreen manuals:
 - *SunScreen 3.2 Release Notes*(PN 806-4129-10)
 - *SunScreen 3.2 Installation Guide*(PN 806-4126-10)
 - *SunScreen 3.2 Administration Guide*(PN 806-4127-10)
 - *SunScreen 3.2 Configuration Examples*
 - *SunScreen SKIP User's Guide, Release 1.5.1*(PN 806-5379-10)
-

How This Book Is Organized

SunScreen 3.2 Administrator's Overview contains the following chapters and appendices:

- Chapter 1 provides a brief overview of the SunScreen product, including operating system and hardware requirements and compatibility.
- Chapter 2 discusses security considerations, administration, security policy, and proxies.
- Chapter 3 explains stateful packet filtering, policy versions, interfaces, administration, security policy, proxies, and rules.
- Chapter 4 describes the components or data objects used in making up the rules for a security policy.
- Chapter 5 describes the types of administration possible, including information on remote administration, local administration, centralized management of groups of Screens, and creating common objects and policies for multiple Screens.
- Chapter 6 describes encryption and decryption, how SunScreen uses encryption, and setting up and using a virtual private network.
- Chapter 7 contains information on NAT rules, static and dynamic NAT, and examples of NAT.
- Chapter 8 describes high availability (HA), developing a high-availability (HA) policy, how HA works, and configuring HA.
- Chapter 9 discusses user authentication, authorized users, administrative users, proxy users, details of RADIUS user authentication, and SecurID authentication.
- Chapter 10 describes SunScreen proxies including how proxies work, proxy user authentication, the FTP proxy, the HTTP proxy, the SMTP proxy, and the Telnet proxy.
- Chapter 11 contains information on packet logging, log file locations, configuring traffic log size, retrieving and clearing logs, log statistics, inspecting and browsing logs, enhancement, and log macros.
- Appendix A contains a table comparing the commands from SunScreen EFS and SunScreen SPF-200 to the equivalent commands used in SunScreen 3.2.

- Appendix B documents the command-line interface.
- Appendix C lists the services and state engines supported by SunScreen.
- Appendix D lists the error messages generated by SunScreen.
- Glossary lists the terms and their definitions used in the SunScreen documentation.

Related Books and Publications

You may want to refer to the following sources for background information on cryptography, network security, and SKIP.

- Schneier, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons, 1996, ISBN: 0471128457
- Chapman, D. Brent and Elizabeth D. Zwicky, *Building Internet Firewalls*, O'Reilly & Associates, 1995, ASIN: 1565921240
- Walker, Kathryn M. and Linda Croswhite Cavanaugh, *Computer Security Policies and SunScreen Firewalls*, Sun Microsystems Press, Prentice Hall, 1998, ISBN 0130960150
- Cheswick, William R. and Steve Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, 1st edition, Addison-Wesley, 1994, ISBN 201633574
- Black, Uyless D., *Internet Security Protocols: Protecting IP Traffic*, 1st Edition, Prentice Hall, 2000, ISBN: 0130142492
- Comer, Douglas E., *Internetworking with TCP/IP*, 3rd Edition, Volume 1, Prentice Hall, 1995, ISBN 0132169878
- Doraswamy, Naganand and Dan Harkins, *Ipsec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks*, 1st Edition, Prentice Hall, 1999, ISBN: 0130118982
- Stallings, William, *Network and Internetwork Security: Principles and Practice*, Inst Elect, 1994, Product#: 0780311078
- Kaufman, Charlie and Radia Perlman, Mike Speciner, *Network Security: Private Communication in a Public World*, 1st Edition, Prentice Hall, 1995, ISBN: 0130614661
- Garfinkel, Simson and Gene Spafford, *Practical Unix and Internet Security*, 2nd Edition, O'Reilly & Associates, 1996, ISBN: 1565921488
- Farrow, Rik, *UNIX System Security: How to Protect Your Data and Prevent Intruders*, Addison-Wesley, 1990, ISBN: 0201570300
- Kaufman, Elizabeth and Andrew Neuman, *Implementing IPsec: Strategies for Making Network and VPN Security Work*, Wiley, John & Sons, Incorporated, 1999

Sun Software and Networking Security: <http://www.sun.com/security/>

A public SunScreen discussion forum at Sun's Support Forum site is also available. See <http://supportforum.sun.com/cgi-bin/WebX.cgi?/security.sunscreen>.

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at <http://www1.fatbrain.com/documentation/sun>.

Accessing Sun Documentation Online

The docs.sun.comSM Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is <http://docs.sun.com>.

Typographic Conventions

The following table describes the typographic changes used in this book.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .login file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	<code>machine_name% su</code> Password:

TABLE P-1 Typographic Conventions (Continued)

Typeface or Symbol	Meaning	Example
<i>AaBbCc123</i>	Command-line placeholder: replace with a real name or value	To delete a file, type rm <i>filename</i> .
<i>AaBbCc123</i>	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in <i>User's Guide</i> . These are called <i>class</i> options. You must be <i>root</i> to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P-2 Shell Prompts

Shell	Prompt
C shell prompt	<code>machine_name%</code>
C shell superuser prompt	<code>machine_name#</code>
Bourne shell and Korn shell prompt	<code>\$</code>
Bourne shell and Korn shell superuser prompt	<code>#</code>

Getting Support for SunScreen Products

If you purchased this product from Sun Microsystems and require technical support, contact your Sun sales representative or Sun Authorized Reseller.

For information on contacting Sun, go to the URL:
<http://www.sun.com/service/contacting/index.html>

For information on Sun's Support go to the URL:
<http://www.sun.com/service/support/index.html>

SunScreen Overview

This chapter describes SunScreen generally and discusses the following topics:

- “What Is SunScreen?” on page 21
- “Software and Hardware Requirements” on page 23
- “Compatibility With Other SunScreen Products” on page 25
- “SunScreen Lite” on page 26
- “Online Help and Documentation” on page 27

What Is SunScreen?

SunScreen 3.2 is a versatile firewall that provides access control, authentication, and network data encryption. SunScreen has a dynamic packet-filtering engine for network-access control (routing mode) and an encryption and authentication engine (stealth mode) for creating secure VPN gateways by integrating public-key encryption technology. SunScreen addresses high availability (HA) for standards-based encryption. Secure administration is provided through a browser-based administration GUI or a command line editor.

Each physical network interface in a SunScreen environment can operate in either routing or stealth mode. An interface in *routing mode* has its own IP address and behaves like a bridge. If two or more routing mode interfaces are present, SunScreen connects multiple networks or subnetworks. Virtual interfaces are supported only in routing mode. An interface in *stealth mode* does not have an IP address, nor does it have a TCP/IP stack. Multiple stealth interfaces act like a bridge that subdivides a single subnet with respect to routing.

SunScreen consists of two primary components: a *Screen* which is the firewall responsible for screening packets and performing necessary encryption and decryption, and an *Administration Station*, where you define your security policy and

from where you administer your Screen or Screens. The two components can be installed on separate machines for *remote administration* or on a single machine for *local administration*.

There are two methods of administering SunScreen: the SunScreen administration graphical user interface (GUI) and the command line configuration editor. Note that the command line interface is a superset of the GUI. The GUI is handy for simple, highly-interactive tasks. The command line interface is scriptable. See Appendix B for detailed descriptions of the configuration editor and the command line interface.

For remote administration of a Screen, you must use either SunScreen SKIP (Simple Key-Management for Internet Protocols) or IPsec/IKE (IP Security Architecture/Internet Key Exchange) for secure communication between the Administration Station and the Screen. IKE or SKIP protects network traffic against unauthorized modification and eavesdropping, and securely authenticates parties that are communicating with each other.

- SKIP technology enables encryption, authentication, access control, and secure virtual private networks (VPN). SunScreen incorporates SunScreen SKIP, release 1.5.1 for the Solaris™ operating environment. You can use the SunScreen administration GUI to administer SKIP on the Screen, but if you need further debugging capabilities, you must use the command line. See the *SunScreen SKIP User's Guide, Release 1.5.1*, for further information regarding SKIP encryption and administration.
- The IPsec protocol is a set of security extensions that use modern cryptographic methods to provide privacy and authentication services. IPsec alone, without IKE, uses manual keying to establish security sessions. SunScreen also supports manual keying. The IKE feature in SunScreen is supported only in the Solaris 8 operating environment. See *System Administration Guide, Volume 3*, for information about IPsec and IPv6. IKE, with signed certificates, must be used for remote administration and for VPNs created with `vpngateway` objects.

Note – For encryption, SunScreen supports IPsec (Internet Protocol Security) with manual keying and IKE (Internet Key Exchange) as well as SKIP (Simple Key Management for Internet Protocol). You can use IKE and SKIP on the same Screen, but they cannot encrypt the same traffic.

You can administer SunScreen remotely from any system that has a browser compliant with JDK 1.1.3 and has a supported version of IKE or SKIP software installed. SKIP software is available for the Sun Solaris operating environment and the Microsoft Windows operating environment (Windows 95, Windows 98, NT 4.x with PC SKIP). IKE is available for Solaris or Windows 2000 with IKE.

Software and Hardware Requirements

The table below lists the installation requirements for SunScreen 3.2.

TABLE 1–1 SunScreen 3.2 Installation Requirements

Requirement	Description
Operating environment	<ul style="list-style-type: none">■ Solaris 2.6, Solaris 7, Solaris 8 (with IPv4 only) in either 32-bit or 64-bit mode for SPARC and the Intel platform editions■ Trusted Solaris 8 (SPARC systems only)
Browsers supported:	<ul style="list-style-type: none">■ A Java™-enabled Web browser compliant with JDK™, Release 1.1.3 or later■ HotJava™ 1.1 running on the SPARC and Solaris Intel platform editions■ Internet Explorer 4.0 (with or without the Java plug-in) on the Windows platform■ Netscape 4.0.1 or higher, can be used for all administrative functions except those requiring local file access. (See below for system requirements for Internet Explorer and Netscape to run Java plug-ins.)
Hardware	<ul style="list-style-type: none">■ All SPARCstation™ workstations, UltraSPARC, and Intel systems supported by Solaris 2.6, Solaris 7, and Solaris 8 operating environments■ All SPARCstations and UltraSPARC systems supported by Trusted Solaris 8
Disk space	<p>Minimum of 1 Gbyte (with at least 300 Mbytes unused). This space is needed for the following:</p> <ul style="list-style-type: none">■ configuration database = /etc/sunscreen = 10 MB¹■ logs and temporary files = /var/sunscreen = 120 MB²■ internal files = /usr/lib/sunscreen = 50 MB■ man pages = /usr/share/man = 1 MB
Memory	<ul style="list-style-type: none">■ For administration software installation: a minimum of 32 Mbytes is required and 64 Mbytes is <i>strongly</i> recommended.■ For Screen-only software installation: a minimum of 32 Mbytes.

TABLE 1–1 SunScreen 3.2 Installation Requirements (Continued)

Requirement	Description
Network interfaces supported	<p>For the Screen:³</p> <ol style="list-style-type: none"> For SPARC and UltraSPARC systems in routing mode: <ul style="list-style-type: none"> 10–Mbps or 100–Mbps Ethernet interfaces (le, qe, hme, be, qfe, pnet) Gigabit Ethernet (ge) interfaces Token Ring interfaces (trp) ATM (155 and 622 Mbps) in LAN emulation mode (lane) or classic IP mode (ba) FDDI (nf), or PCI-based Ethernet cards For SPARC and UltraSPARC systems in stealth mode: 10–Mbps, 100–Mbps, Fast Ethernet, or Gigabit Ethernet interfaces For Intel-based systems: 10 Mbps or 100 Mbps Ethernet interfaces (dnet, elx1) High availability requires that the two machines be connected by means of a non-switching hub.⁴ <p>For the Administration Station:⁵</p> <ol style="list-style-type: none"> For SPARC systems: 10–Mbps or 100–Mbps Ethernet interfaces (le, qe, hme, be, qfe), or FDDI, or PCI-based Ethernet cards. An Administration Station can connect to the Screen by an asynchronous transfer mode (ATM) or Token Ring LAN, but only after it is connected directly to the network by way of an Ethernet or FDDI connection first. For Solaris Intel Edition systems: 10–Mbps or 100–Mbps Ethernet interfaces (dnet, elx1).
Media	CD-ROM drive (and a diskette drive, if you are using certain types of CA-issued certificates).

- Can grow larger over the course of hundreds of policy or configuration changes
- Can grow larger if the SunScreen log size parameter is increased from its default of 100 MB
- The Screen can support up to 15 stealth interfaces at one time.

Stealth configurations do not support ATM, FDDI, token ring, or the use of proxies.

SunScreen HA in routing mode does not support FDDI, token ring, ATM, Gigabit Ethernet, or failover of IKE-based IPsec connections
- Some switches, including Alteon, Radware's Fireproof, and Foundry's ServerIron, can be configured to work with SunScreen HA clusters. Each Screen is set up as an individual Screen, with different IP addresses, and no interconnect. You can use as many Screens as the switch supports. Note that because SunScreen is a stateful firewall, TCP connections do not failover.
- A remote Administration Station can connect directly to a Screen only through an Ethernet local area network (LAN) or a fiber distributed data interface (FDDI).

SunScreen includes HotJava 1.1, SunScreen SKIP for Solaris, and IKE software.

Note – To read the SunScreen documentation from the administration GUI, you must have the Adobe Acrobat Reader plug-in installed on your system.

Note – Because of a limitation in SunScreen SKIP 1.5.1 for Solaris, the RC2 encryption algorithm is not available when running Solaris 7 or 8 in 64-bit mode.

Required Patches

See the *SunScreen 3.2 Installation Guide* for a list of required patches.

Java Plug-In Software

With Java plug-in software applets using Java technology on your Web pages can use Java Runtime Environment (JRE) instead of the browser's default runtime. Java plug-in software is available for Microsoft Windows and Sun Solaris-based browsers.

Java plug-in software system requirements:

- Windows 95, Windows 98, or Windows NT 4.0
 - Pentium 90 MHz or faster processor
 - 10 MBytes free hard disk space (recommended 20 MBytes)
 - 24 MBytes system RAM
- Solaris 2.5 or compatible versions
 - SPARC or Intel microprocessor
 - 10 MBytes free hard disk space (recommended 20 MBytes)
 - 32 MBytes system RAM (recommended 48 MBytes)

Java plug-in software is available at no charge at the following URL:
<http://java.sun.com/products/plugin/1.1.2/index-1.1.2.html>

See Appendix A, "Using the Command Line," in the *SunScreen 3.2 Administration Guide* for instructions on how to install the plug-in software.

Compatibility With Other SunScreen Products

SunScreen 3.2 can communicate with older SunScreen firewalls either in the clear or as part of a VPN.

The obsolete `ss_client` command that is used in SunScreen SPF-200, Release 1.0, and SunScreen EFS, Release 2.0, is maintained so that you can still manage Screens running these versions of the software remotely through the command line.

Note – See Appendix A for information regarding command compatibility with previous releases. For information regarding the current commands for SunScreen, see Appendix B.

The SunScreen SKIP encryption system built into SunScreen 3.2 is compatible with other SKIP implementations, such as earlier releases of SunScreen firewall products, *SunScreen SKIP for Solaris*, or *SunScreen SKIP for the Microsoft Windows Operating Environment*. SunScreen 3.2 exchanges encrypted information with other SunScreen firewall products transparently.

To upgrade to SunScreen 3.2 from earlier SunScreen firewall releases, see the upgrading instructions in the *SunScreen 3.2 Installation Guide*.

SunScreen Lite

SunScreen Lite is a stateful, packet-filtering firewall that has a subset of the features in SunScreen. It protects individual servers and small work groups.

This manual is a reference for both SunScreen Lite and SunScreen applications. Keep the following differences and similarities in mind when configuring and administering SunScreen Lite.

Supported Features

The SunScreen 3.2 Lite firewall:

- Can be administered from a remote Administration Station.
- Supports basic packet filtering.
- Displays all data for supported SunScreen types and data fields.
- Can be used in virtual private networks (VPNs).
- Can be used for secondary machines in a centralized management group.
- Uses SunScreen SKIP or IKE for encryption. SunScreen SKIP and IKE are included as part of SunScreen 3.2 Lite and are automatically installed.

Limitations

The SunScreen Lite firewall does not support some features that are available in SunScreen. A SunScreen Lite firewall:

- Does not support stealth-mode operation.
- If you have more than two interfaces and `ip_forwarding` is on, cannot support more than two routing interfaces. Any additional interfaces that are configured on this system will not have filtering rules applied to them. Note that Lite supports virtually unlimited routing interfaces when the Screen is *not* acting as a router—when `ip_forwarding` is turned off. This is ideal for protecting server systems that have multiple interfaces for connectivity, administration, and backup, but that are not routing packets between interfaces.
- Does not support and cannot create the ADMIN, HA, or STEALTH interfaces.
- Cannot support more than ten unregistered IP addresses that can be translated to registered addresses using network address translation (NAT); it is limited to two NAT rules.
- Cannot be a member of a high availability (HA) cluster.
- Cannot create and cannot be made the primary Screen in a centralized management group (CMG).
- Does not support proxies.
- Does not support and cannot create time objects.
- Ignores the time-of-day field in rules.

Online Help and Documentation

Topical help is available for each page of the administration GUI by clicking the Help button on a page or by clicking the Documentation button.

Click the Documentation button for the PDF files. They are installed with the `SUNWSfwd` package.

The man pages for SunScreen are located in `/usr/share/man/man4sunscreen`.

The man pages for SunScreen SKIP and IPsec/IKE are located in the standard Solaris man page directory.

SunScreen Concepts

This chapter explains why SunScreen is important for protecting your company's proprietary information and describes how SunScreen works. It includes the following topics:

- "Why Use SunScreen?" on page 29
- "How SunScreen Works" on page 30
- "Routing and Stealth Mode Interfaces" on page 32
- "Administration" on page 33
- "Security Policy " on page 36
- "Proxies" on page 40
- "IPsec/IKE" on page 41

Why Use SunScreen?

Your company may want to provide Internet services for customers and others, while allowing your employees to connect to the Internet for services and for access to corporate information. However, such connections may put your company assets at risk. SunScreen protects your company's assets by inserting a firewall between them and the Internet.

A Sample SunScreen Network Map

SunScreen divides the network into discrete areas, each served by an interface. You set up filtering rules to control access to one area from another area, which can be another network within your company or an area outside your company.

The following figure shows a sample map of a simple network in which a Screen in stealth mode functions as a firewall to connect the Engineering network over an unsecured public network (the Internet) through a Screen in routing mode to other secure networks.

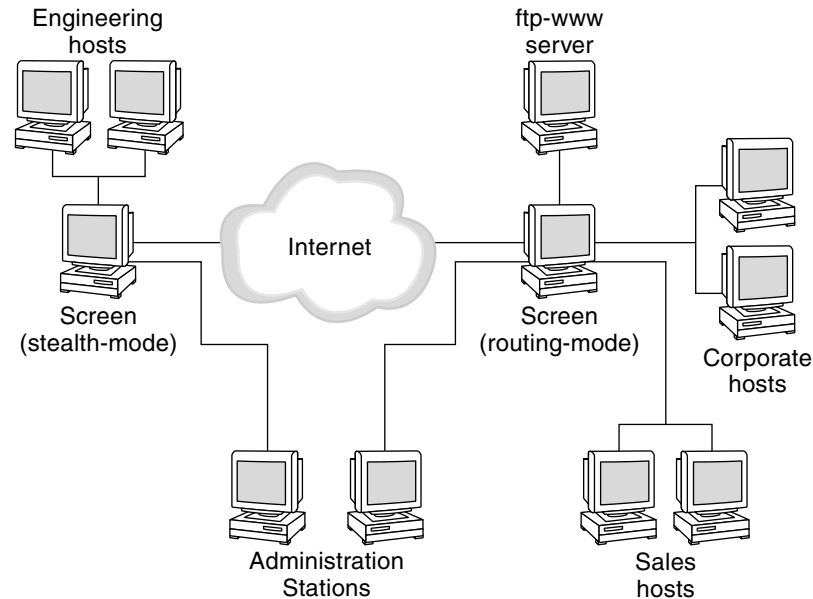


FIGURE 2-1 Sample Network Map

The ftp-www server might be the *public* area of the company, also called the *demilitarized zone* (DMZ), and the engineering, sales, and corporate network segments might be part of the *private* area. SunScreen can then control access between these areas and the rest of the Internet.

See “Defining Security Policies” in *SunScreen 3.2 Installation Guide* for worksheets and instructions to aid you in determining your network configuration and your desired security level.

How SunScreen Works

SunScreen is a Solaris software product that supports the following platforms: Solaris 2.6, Solaris 7, and Solaris 8 and compatible versions, for both the SPARC and Intel platforms, and Trusted Solaris 8 for the SPARC platform.

Note – Upgrade your system to at least Solaris 2.6. SunScreen cannot be used with Solaris 2.5.1 because of Unicode internationalization requirements.

For local administration, the administration GUI works on any hardware or software system with a browser that supports JDK 1.1 (up to and including 1.1.3). For remote administration, the administration GUI works on any hardware or software system with a browser that supports JDK 1.1 (up to and including 1.1.3) and SKIP or IPsec/IKE installed.

Integration of the two SunScreen firewall products—SunScreen EFS and SunScreen SPF-200—in SunScreen 3.2 enables you to create:

- A stealth-mode firewall as a dedicated perimeter defense and extranet firewall
- A routing-mode firewall as a traditional firewall on the perimeter of a network
- A remote-access server inside the intranet to segregate departments

You can also deploy SunScreen on existing application and data servers throughout an enterprise to control access and provide encryption.

SunScreen includes the following graphical user interfaces (GUIs):

- SunScreen install wizard
- SunScreen administration GUI
- SunScreen SKIP `skiptool` GUI (for configuring SKIP on an Administration Station)

Use the install wizard to configure your Screen in routing mode (the default) or in stealth mode.

Note – For machines without a monitor, use `pkgadd` to install SunScreen and `ssadm configure` to configure the initial configuration.

Following installation, use the administration GUI to administer your Screen locally (on the same machine) or remotely (from an Administration Station). With the administration GUI you can administer single Screens, high availability (HA) clusters, or centralized management groups (CMGs) of Screens—locally or remotely.

Note – The `skiptool` GUI is not available on a SunScreen Screen; it is incompatible with SunScreen's SKIP implementation.

The behavior of a SunScreen is governed by a set of ordered rules called a *policy*. Individual policies and versions or histories of a policy can be copied or saved into a

new policy. Each version of a policy is stored in a separate file on the (master) Screen. You can either use all of a policy or a portion of a policy at a later date.

The network address translation (NAT) feature enables the Screen to map unregistered internal network addresses to a registered network address. NAT can also map an internal network address to a different network address. As the packet passes between an internal host and a public network, it replaces the addresses in the packet with new addresses transparently, corrects the checksums and sequence numbers, and monitors the state of the address map. You specify when the ordered NAT rules apply to a packet based on source and destination addresses. Note that SunScreen performs NAT of source address before encrypting and NAT of destination address after decrypting. See Chapter 7 for detailed information about NAT.

High availability (HA) protects data by using a set of Screens to provide failover protection. One member of the HA cluster, the active Screen, services packets travelling between a protected inside network and an insecure outside network. Other members—the passive Screens—receive the same packets, perform the same calculations, and mirror the state of the active Screen, but they do not forward traffic between the inside network and the outside network. One of the passive Screens automatically becomes the active Screen if the active Screen fails.

Routing and Stealth Mode Interfaces

SunScreen includes routing-mode capabilities from SunScreen EFS and stealth-mode from SunScreen SPF.

Routing Mode Interface

Routing-mode interfaces have IP addresses and can perform IP routing. Routing mode requires that you connect each interface to a different network with its own network number.

Access to all proxies is through the transmission control protocol (TCP) and can only run on Screens configured with at least one routing mode interface.

Stealth Mode Interface

A SunScreen in stealth mode bridges the MAC layer. It therefore partitions an existing single network and, consequently, does not itself divide the network into subnetworks. A stealth-mode interface does not have an associated IP address.

In stealth mode, you must configure one interface as an administration interface (to perform remote administration). This interface is special case of a routing interface that is configured so that it only passes encrypted administration traffic between the Screen and a remote Administration Station.

Hardening the OS

If all of your filtering interfaces are in stealth mode, SunScreen offers optional *hardening* of the OS, which removes packages and files from the Solaris operating system that are not used by SunScreen—in accordance with the best practices as described in

<http://www.sun.com/blueprints/browsesubject.html#security>.

Hardening in SunScreen 3.2 is based upon JASS (JumpStart Architecture and Security Scripts). The JASS scripts are in `/usr/lib/sunscreen/admin/jass`. The hardening script is `/usr/lib/sunscreen/lib/harden_os`. The process of hardening can be carried out at install time or at a later time by running the script.

WARNING: this script cannot be reversed; once files have been removed, the only way to recover them is to reinstall Solaris.

Note – If some of your filtering interfaces are in stealth mode and other interfaces are in routing mode, you should not use the option of hardening of the OS that SunScreen offers.

Administration

SunScreen consists of two components: an *Administration Station* and a *Screen*. The two components can be installed on separate machines with Screen on one or more machines and another machine as a remote Administration Station, or they can be installed on a single machine for local administration of a Screen. If both components are installed on a single machine, the Administration Station can administer not only the local Screen, but other Screens that are remote as well.

The number of Screens and Administration Stations needed at a site depends on its network topology and security policies. Typically, one Screen is installed at each

network-direct public access location that needs to be restricted. An Administration Station can manage multiple Screens.

You typically choose whether to administer a Screen locally or remotely when you install the SunScreen software. Alternatively, you can add a remote Administration Station after the Screen software has been installed.

Local Administration

Local administration is performed on the host where the Screen software is installed, as shown in the figure below. Because administrative commands do not travel over a network, local administration does not require encrypted communication.

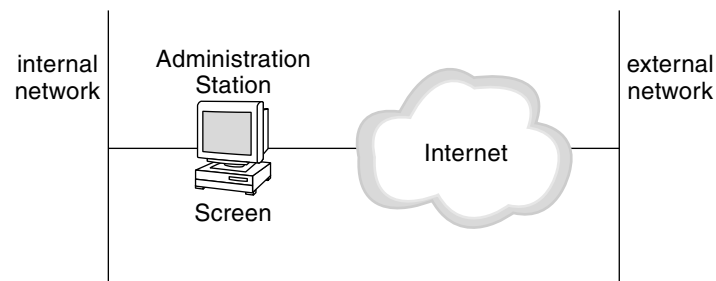


FIGURE 2-2 Local Administration of a Screen

Remote Administration

Remote administration is performed from an Administration Station, where the administration software, including SunScreen SKIP and/or IKE, is installed. As shown in the figure below, a remote Administration Station on the internal network administers the Screen located between the internal network and the Internet. This Screen could be configured as the router between the internal network and the Internet. A second remote Administration Station for this Screen is located on the external network. The Administration Stations must be configured to communicate with the Screen using encryption. SunScreen SKIP or IKE is used to encrypt all communication between the remote Administration Stations and Screens, regardless of network topology between them.

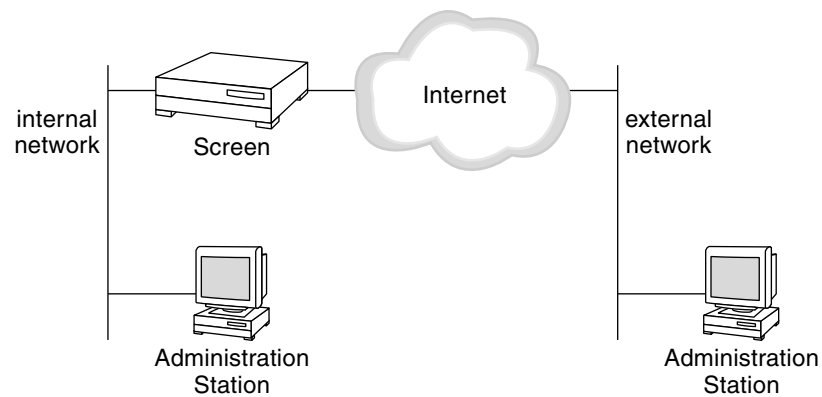


FIGURE 2-3 Remote Administration From an Administration Station to a Screen

Locating the SunScreen Screen

A Screen can only control traffic that passes through it; the Screen must, therefore, be placed in the network such that the traffic you want to control passes through it. All packets coming into the network and leaving it must pass through the Screen that controls the network.

If multiple paths exist between the Internet and the corporate network, the Screen will not work optimally, because, depending on the routing, traffic can pass through the Screen in one direction, but can bypass it in the reverse direction. To control the traffic on a network properly, both incoming and outgoing traffic must pass through the same Screen.

The figure below shows a network divided into several pieces by a Screen.

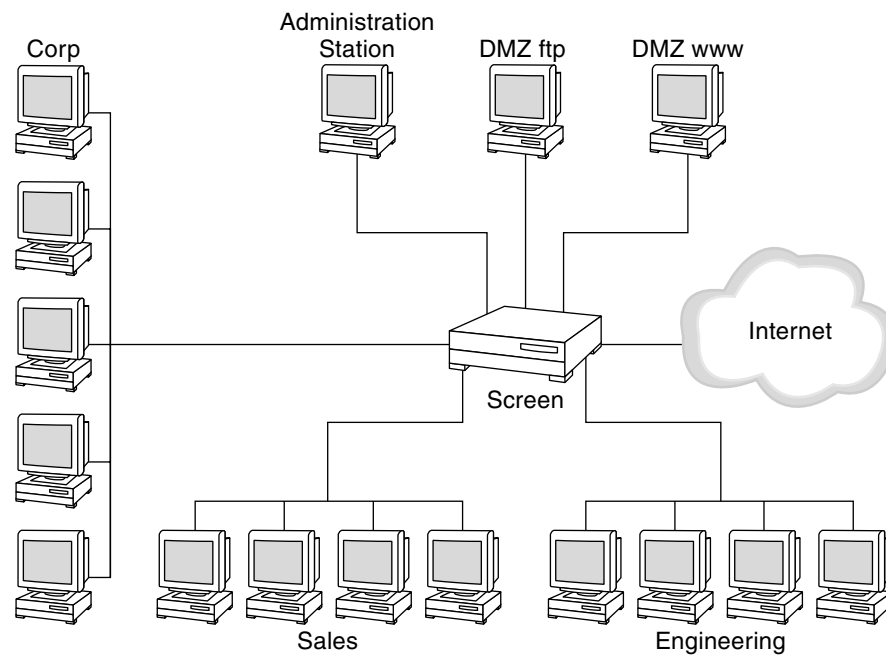


FIGURE 2-4 SunScreen as Internet Firewall Dividing a Network into Several Pieces

In the figure above there are two networks: the Internet and the company's network. The company's network is further divided into several demilitarized zones (DMZs) where public services reside. The advantage of dividing the network into two or more DMZs is that even if a system on a DMZ is compromised, the traffic on that system must still pass through the Screen.

Security Policy

A security policy is the collection of decisions an organization makes about network security and its stance regarding what network activities are permitted or denied. The most important aspect in installing and administering a firewall is a well-defined security policy.

Configuration

A *SunScreen policy* comprises the rules that a SunScreen Screen uses to implement your company's security policy. A *configuration* is the union of a SunScreen policy with common objects to form a complete description of the behavior of one or more Screens. A *policy* is a named set of policy objects. When the SunScreen software is first installed, there is one policy, named *Initial*, which contains a single rule and objects for the Screen and its interfaces. Common objects are data objects relevant to all policies. Common object types include address, screen, service, interface, certificate, and time. Ordered objects include filtering rules, NAT rules, administration access rules, and VPN (virtual private network) gateway descriptions.

Neither common objects nor rules include objects loaded into SKIP or IKE, but they do include the reference from the certificate name in the common object registry to the internal identity used by SKIP or IKE.

Stateful Packet Filtering

A Screen, which sits between the client and server, uses *stateful packet filtering* to examine each data packet as it arrives. Based on information in the packet, state retained from previous events, and a set of the security policy rules, the Screen either passes the data packet or blocks it.

SunScreen uses a set of ordered rules to filter packets. When you configure SunScreen, you translate the security policies for your site into a series of policy rules that specify which services are allowed, what to do with packets for services that are disallowed, and what to do when packets are dropped. You then place these policy rules in sequence to specify which rules override others.

Centralized Management Group

A centralized management group (CMG) reduces the overhead in configuring a set of firewalls and enables you to locate Screens at different points in the network and group them as an object. You can then manage this group of Screens with a standard set of objects through an Administration Station. All the firewalls in the group share the same policy, but apply it based upon their location in the network topology.

Network Address Translation (NAT)

Network address translation (NAT) translates one set of IP addresses to another set. NAT is typically used to:

- Hide the internal topology of a network – When the network’s private (unregistered) addresses are translated to a set of public (registered) addresses, all traffic appears to come from this set of public addresses rather than from the network’s private addresses.
- Use the public addresses assigned to a site efficiently – Many sites have a limited set of registered addresses assigned to them by their Internet service provider (ISP). With NAT you can use those addresses efficiently to support a large internal network. In this case, the addresses of the internal network are translated to the smaller set of registered addresses assigned by the ISP.
- Prevent having to renumber host addresses when changing ISPs – NAT enables you to map your old public addresses to the new set of registered addresses assigned by your new ISP.
- Use private or unregistered addresses on your internal network – The Internet Assigned Number Authority (IANA) has reserved the address ranges 10.0.0.0 through 10.255.255.255, 172.16.0.0 through 172.31.255.255, and 192.168.0.0 through 192.168.255.255 for use in private networks. NAT enables you to use these addresses in your internal network, yet still communicate with other networks by mapping those internal addresses into the registered addresses assigned to your site.

NAT modifies the address fields in the IP header of the packet as it passes through the Screen. It also modifies the checksum and sequence number fields in the packet. Certain protocols (such as `ftp`) also require that data within the packet containing address information be modified.

Tunneling and Virtual Private Networks (VPN)

Organizations typically have offices in more than one location. SunScreen provides a *tunneling* mechanism to let the different offices use public networks as a secure private network without needing dedicated lines and with no changes to user applications.

When a tunnel, or virtual private network (VPN), is set up between two or more locations, all data packets traveling from one location to the other are encrypted and wrapped inside other packets before they are sent over the public Internet. Encrypting the packets guarantees that their contents will remain private; anyone capturing packets with the snoop program on network traffic between the two locations will be unable to read them. When the packets arrive at the remote location, they are unwrapped, decrypted, and forwarded to their intended destination.

In addition to protecting the privacy of network traffic, tunneling also lets a site conceal the details of its network topology from intruders or eavesdroppers. Because the original packets are encrypted, the source and destination addresses in their IP headers cannot be read. When the encrypted packets are encapsulated inside other packets, the new IP headers identify the addresses of the Screens that protect the

locations, not the hosts that originated the packets. Consequently, the network topology behind the Screens is never exposed.

High Availability (HA)

High Availability (HA) enables you to deploy groups of Screens together in situations in which the connection between a protected inside network and an insecure outside network is critical. At any time, one member of the HA cluster is the *active* Screen, which performs packet filtering, network address translation, logging, and encryption or decryption of packets travelling between the inside and outside networks. The other members of the Screens, receive the same packets, perform the same calculations as the active Screen, and mirror the state of the active Screen, but they do not forward traffic. When an active Screen fails, the passive Screen that has been running the longest takes over as the active Screen within 15 seconds. During this time (before the passive Screen takes over), no traffic will go through the HA cluster.

HA cluster, the *passive*

SunScreen provides flexible logging of packets. This means that each primary and secondary Screen can keep a log of its traffic. Logs of the packets are kept on the Screen that passed or rejected the packets.

Encryption

SunScreen uses a combination of public-key and shared-key cryptography to encrypt and decrypt packets. Any traffic that passes between any two machines or other SKIP or IKE devices can be encrypted. All traffic between a Screen and an Administration Station is encrypted.

Logging

You can configure SunScreen to log a packet when it matches a rule or when it does not match any particular rule. Most frequently, packets matching DENY rules or packets that are dropped because they do not match any rule are logged. The action defined in a rule controls whether a packet is logged and what information about the packet is recorded.

Examining logged packets is useful when you are trying to identify the causes of problems during configuration or administration. You should also examine logs periodically for evidence of attempts to break into your network.

In an HA cluster, only the active Screen logs network traffic. However, traffic destined for the active or passive HA machine itself may be logged according to the rule. This means that some passive Screens may log some traffic, but only the traffic to them, not the traffic that is going through them. Each machine in an HA cluster logs what that system passed or rejected, as well as any locally processed nonpacket events.

Proxies

A proxy is a user-level application that runs on the Screen. The main purpose of proxies is to provide content filtering (for example, allow or deny Java applets) and user authentication.

You can set up proxies for `ftp`, HTTP, SMTP, and Telnet protocols. Although each proxy has different filtering capabilities and requirements, you can allow or deny sessions based on the source or destination address of packets. Proxies share common objects and policy rule files. To start a proxy, you set up rules for a proxy in your security policy and activate the policy.

Use of these proxies does not require installing any additional client or server system software. Some changes, however, may be required in system configurations or user-supplied commands to have access to protected destinations through the proxies.

Event Logging With Proxies

In addition to packet (SUMMARY and DETAIL) and SESSION log events, authentication, editing and activation, HA, proxies and other components of SunScreen create additional log entries. These entries are added to the SunScreen log chronologically, interspersed as events take place. Events such as administrator or proxy user authentication (or denial), policy-related events, HA failovers, proxy connection establishment, connection completion, transfers, as well as large a variety of debugging messages are inserted.

These non-packet, non-session events are sometimes referred to as extended events, since the mechanism used to log them is both an extension of earlier types of logging *and* it is itself extensible. They consist almost entirely of printable (UTF-8) strings

These extended events are flagged by `ssadm logdump` as XLOG. In the log browser, they are flagged with the severity level, followed by the application name.

Extended log events are described in detail in Chapter 11.

IPsec/IKE

IPsec is the name of the IETF standard system for Internet Protocol security. Specified in RFCs 2401, 2402, 2406, and 2409, among others, IPsec provides network-layer security for IP packets, and therefore, for any protocol that is based on IP. IPsec provides approximately the same functionality as SKIP. Both were developed from the same ideas and technology, but IPsec has been adopted as an IETF standard and is supported by a growing variety of software and equipment vendors. IPsec security can include:

- Confidentiality – An unauthorized party can not observe any of the protected data
- Integrity validation – An unauthorized party can not modify or damage protected data except by causing entire packets to be lost or delayed
- Authentication – An unauthorized party cannot impersonate the legitimate sender of data

When a packet is sent using IPsec, it contains security information in addition to the normal payload and IP headers. If confidentiality is desired, the normal payload and some of the headers are encrypted using a cryptographic key, and can only be decrypted by a recipient with the correct key. If authentication and integrity protection are desired, a message authentication code (MAC) is computed by the sender using a cryptographic key and can be verified by a recipient with the correct key.

The security information in an IPsec packet is carried in one or both of the special headers that form the IPsec protocols: AH and ESP. AH is an authentication header inserted between a packet's IP header and its transport layer payload. AH does not modify the payload; it merely adds an integrity check value so that a recipient can confirm the authenticity of the payload. ESP (encapsulating security payload), on the other hand, does modify the payload by encrypting it, making it unreadable until a recipient decrypts it using the correct key.

A variety of encryption and authentication algorithms can be used, but both the sender and receiver must use the same algorithms and keys. Multiple "flows" of packets using different algorithms or keys can exist simultaneously between a pair of systems. Each flow of packets using a particular algorithm and key is identified by a security parameters index, or SPI, which is included in the IPsec headers for the receiver's use. The association of an algorithm, a key, and an SPI with a packet flow is called a security association (SA). Each system using IPsec has a security association database that helps control the processing of incoming and outgoing IPsec packets. A security association is unidirectional, so it is typical to have a pair of SAs, one in each direction, for bidirectional protocols such as TCP.

The specific algorithm and key used by each SA can be configured into both systems manually, or can be determined by an automatic process called the Internet Key Exchange (IKE). To configure an SA manually, each administrator of two

communicating systems enters the SPI, algorithm, and key identically on each system. To use IKE, the administrator enters the algorithm to be used; IKE automatically chooses keys and changes the keys periodically to minimize the risk of massive data compromise in the event that a key is stolen or "cracked." Another difference between manual keying and using IKE is that IKE chooses parameters and sets up SAs for both directions of traffic, while the manually configured SAs have to be explicitly set up in each direction.

Internet Key Exchange (IKE)

IKE defines a two-phased protocol to establish IPsec SAs for the communication peers. In Phase I, an authenticated, ephemeral Diffie-Hellman exchange allows the peers to authenticate each other, and provides a secure communication channel (an IKE security association) for later use. The IKE SA created in Phase I is then used in Phase II to establish the IPsec SAs used in the ESP and AH security protocols. Multiple Phase II exchanges can be performed under a protection of a single Phase I IKE SA.

One of the following authentication methods must be chosen for the two IKE peers to authenticate each other:

- Pre-shared key
- DSS signatures
- RSA signatures
- RSA encryption

IKE supports the use of X.509 public key certificates for authentication. SunScreen maintains a certificate database for the local system's certificates and for certificates of IKE peers with which the system may communicate. Certificates can be generated on a SunScreen system itself—self-signed, and manually distributed and verified—or can be generated and signed by a certificate authority, the signature validated automatically using the certificate authority's public certificate. An alternative to certificates is the use of a pre-shared key by both peers to authenticate each other, but this method has some security disadvantages.

Other security parameters that must be chosen for Phase I include an encryption algorithm (SunScreen supports DES and Triple-DES), an authentication algorithm (SunScreen supports MD5 and SHA-1), and an Oakley Group (SunScreen supports groups 1, 2, and 5).

Note – RSA-ENCRYPTION cannot use certificate payload. This precludes interoperating with Win2K using this authmethod. Since Win2K does not support this authmethod, this is not a significant problem. The following two conditions must be met for RSA-ENCRYPTION:

- The certificate must have a Subject Alternative Names IP field.
 - The certificate RSA modulus should be at least 1024 bytes—to ensure that it is large enough for the subject name.
-

SunScreen IPsec Configuration

SunScreen's IPsec module uses the same cryptographic (encryption and authentication) modules as the Solaris IPsec implementation. The IPsec encryption and authentication algorithms must be installed and configured in the Solaris kernel in order to be used by SunScreen. SunScreen and Solaris support DES and Triple-DES for encryption and MD5 and SHA-1 for authentication. You may need to install the optional software packages `SUNWcyr` and/or `SUNWcyrx` to make these algorithms available.

Configuration of IPsec and IKE in SunScreen is done through parameters to the rules. Please refer to `ssadm-edit(1m)` and `rule(4sunscreen)`. Commands are provided for certificate generation and certificate database maintenance; see the man pages for `ssadm-certlocal(1m)` and `ssadm-certdb(1m)`.

Packet Filtering

This chapter describes a possible SunScreen configuration and explains what packet filtering is. It includes the following topics:

- “SunScreen Model” on page 45
- “Stateful Packet Filtering” on page 46
- “Policy Rules” on page 48
- “Policy Versions” on page 52

SunScreen Model

When a packet passes from system A to system B, the following takes place:

1. System A looks in its routing table for a route to system B and finds one through the firewall.
2. System A must send the packet to the firewall for routing to system B. It sends out an ARP (address resolution protocol) request and receives a reply from the firewall. System A now knows the MAC address of the firewall and sends the packet to the firewall.
3. The firewall takes the packet and compares its source IP address, destination IP address, and service against the rules. If the rules permit passing the packet, a state-table entry is created and the packet is passed to the routing interface nearest system B (routed).
4. The firewall sends out an ARP request to get the unique MAC (media access control) address of system B; the packet is sent to system B.
5. If this packet is part of a *session* that requires packets to flow back to system A from system B, they are passed because the state-table entry created allows this. In an HA configuration, all members of the HA cluster must have the same state-table

entries. If they do not, sessions can be dropped if the active Screen fails and a passive Screen becomes the active Screen.

Stateful Packet Filtering

A Screen, which sits between the client and server, uses *stateful packet filtering* to examine each data packet as it arrives. Based on information in the packet, state retained from previous events, and a set of security policy rules, the Screen either passes the data packet, or blocks and drops it.

SunScreen uses a set of ordered rules to filter packets. When you configure SunScreen, you translate the security policies for your site into a series of policy rules that specify which services are allowed, what to do with packets for services that are disallowed, and what action to take when packets are dropped. You then place these policy rules in sequence to specify which rules override others.

When the Screen receives a packet, it tests the packet against the rules in a policy in order. The Screen does not need to test each packet against each rule; it assumes that the first rule to match the service, source address, and destination address of the packet is the rule that governs the disposition of the packet. Additionally, if the applicable rule so specifies, the Screen records a log message or generates an SNMP (simple network management protocol) alert.

Note – Spoof detection and checking against existing connections occurs before policy rules checking.

If the packet does not match any rule, the Screen uses its default action for the interface on which the packet arrived to determine the disposition of the packet. Typically, this action logs the packet and drops it; other actions such as logging and emitting an ICMP or SNMP message are available options.

The sequence in which a Screen performs packet filtering, network address translation, and encryption and decryption depends on the direction in which the packets are moving, for routing mode as shown in Figure 3–1.

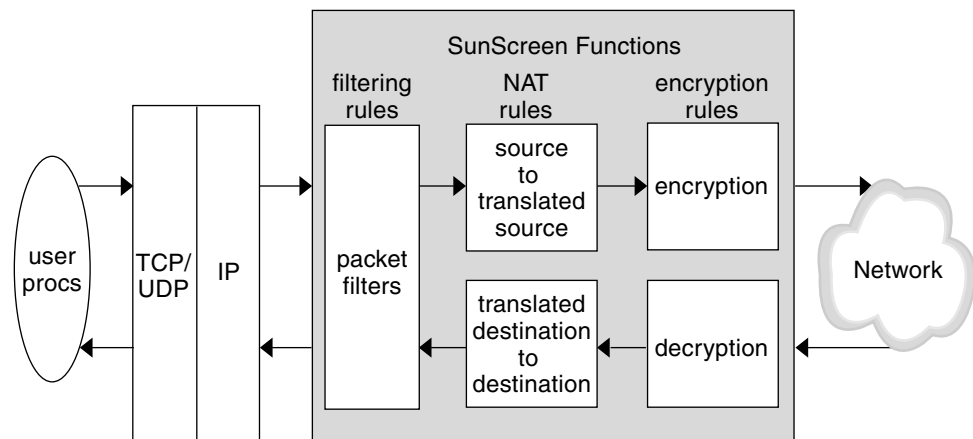


FIGURE 3-1 Rule Processing Order in a Routing Mode Screen

Inbound Packet Rule Checking

When a packet arrives on *any* of the defined interfaces it is:

1. Decrypted if it has been encrypted with the Screen's public key. The Screen calls the SKIP key manager or IKE, which identifies the appropriate keys and decrypts the packet. The SKIP key manager or IKE passes the packet back to the Screen.
2. Translated, if necessary. The Screen determines whether it should use NAT to convert the packet's destination IP address to an internal address. If NAT is used, the destination address information in the packet is changed to the translated destination address.
3. Filtered according to the packet-filtering rules. The Screen applies its filtering rules to the packet to determine whether the packet meets the criteria of an ALLOW or DENY rule. If the packet meets the criteria of an ALLOW rule, the Screen forwards the packet. If the packet meets the criteria of a DENY rule, it is denied. If so specified, the Screen can create a log entry identifying the circumstances.

A packet is only tested against the packet-filtering rules once.

Outbound Packet Rule Checking

If the packet is allowed by the rules (and is not destined for the Screen itself), before being sent out on the appropriate interface, it is:

1. Translated – The Screen determines whether it should use NAT to convert the source address information in the packet. If NAT is being used, the translated

source address information is modified to the source address. Depending on the type of packet, the Screen may also modify address information in the data portion of the packet or modify packet checksums.

2. Encrypted – The Screen determines whether the packet should be encrypted based upon the information in the rule that matched. If the packet should be encrypted, the Screen calls the SKIP key manager or IKE, which identifies the appropriate encryption keys and algorithms and encrypts the packet. The SKIP key manager or IKE passes the encrypted packet back to the Screen, which then forwards the packet.

A packet is only tested against the packet-filtering rules once.

Policy Rules

You can define rules and order them either from the administration GUI or the command line editor.

You set up *ordered policy rules* to reflect the security policy for your site. These rules specify the action to be taken for services between two addresses that are on different interfaces of the Screen. A collection of rules forms a policy.

When you install SunScreen, you start with a policy called `Initial` that establishes access rules for basic TCP/IP services. You then define policy rules to allow clear or encrypted communication between hosts that meet your criteria.

Each rule must specify all three selection criteria: source address, destination address, and type of service. Each rule can also specify what action to take when a packet meets those criteria. For example, different rules will specify whether the Screen should forward or drop the packet; encrypt or decrypt the packet; record the packet in the Screen logs; and issue ICMP messages or SNMP traps concerning the packet. The default rule is to drop any packet that does not have a specific VPN, ENCRYPT, or ALLOW action.

Sequence of Rules

The sequence in which rules are ordered is critical. When the Screen processes packets, it compares the packet information to each rule in order. When a packet meets the criteria of a rule, the Screen applies the actions specified for that rule and disregards the following rules.

Define services for a service group carefully and place the rules in an order that will permit the services used to pass properly. In general, services which require more sophisticated protocol knowledge must appear before or within a rule which would allow the same traffic through a less-sophisticated service definition. If you want to deny FTP packets, for example, you must place the rule denying FTP packets before the more general rule that allows common services:

```
1. ftp * * DENY
2. common services * * ALLOW
```

If you put the `common services` rules before the `ftp` rule, FTP packets will be allowed because `ftp` is one of the common services.

For a more complex example, consider the case of a service group such as `tcp_all`, which does not include services such as `ftp`, `rsh`, and `realaudio`. If you want to deny FTP packets and a `tcp_all` rule occurs before the `ftp` DENY rule, the `tcp_all` rule will pass the `ftp` packet, because it assumes a simple `tcp` connection. Other special-case processing, such as a reverse data-port connection from the server back to the client, will not automatically be allowed by the initial connection as it would had the `ftp` rule been placed first.

Services and Service Groups in Rules

When a packet filter checks to see if a packet matches a rule, the difference between a service or service group such as `ip_all`, and a service or service group with `BROADCAST`, such as `ip_all_with_broadcast`, is important. If the service or service group has the `BROADCAST` flag set, the destination IP address in the packet *must* be a recognized broadcast address.

For example, if services such as `ftp`, `rsh`, and `realaudio` are not part of a service group such as `tcp_all`, and a `tcp_all` rule occurs first, the `tcp_all` rule will pass the `ftp` packet, because it assumes a simple `tcp` connection. Other special-case processing, such as a reverse data-port connection from the server back to the client, will not automatically be allowed by the initial connection as it would had the `ftp`-based rule been placed first.

Configure stealth mode to identify the network and netmask that the Screen partitions so that it can correctly identify what the valid broadcast addresses are. If a packet must pass through the Screen that has a destination address of one of the broadcast addresses, set `BROADCAST` in the service used in the rule to pass the traffic. With respect to all other destination addresses, `ip_all` and `ip_all_with_broadcast` are identical.

Note – When you define rules for specific services (such as telnet, HTTP proxy, ftp, etc.), you do not automatically pass network control messages that might be necessary for these rules to work correctly. You may need to add additional rules that allow this traffic to pass. For example, you can add the `icmp_all` rule that allows all internet control messages to pass.

Rule Syntax

See the *SunScreen 3.2 Administration Guide* for procedural information about creating SunScreen policies with the administration GUI. The basic command-line syntax for a rule is:

Service Source_address Destination_address "action" (ALLOW or DENY) plus optional fields: Time Screen Comment Encryption Proxy User VPN

See "add rule" in the man page for `ssadm-edit(1M)` and "add rule" on page 251 for more information about command line rule syntax.

- *Service* – An individual network service or a group of services provided to users. Sample services are `smtp`, `rlogin`, and `ftp`.
- *Source_address* – An individual network address, a range of addresses, or a group of addresses. Named addresses are also used to define the network interfaces. "Source" refers to the network address from which the packet is coming.
- *Destination_address* – An individual network address, a range of addresses, or a group of addresses. Named addresses are also used to define the network interfaces. "Destination" refers to the network address to which the packet is going. See "Address Object" on page 60 for more information.
- *"action"* – What the Screen should do with packets. The action choices are `ALLOW` (default) and `DENY` (only `ALLOW` is valid for encryption actions; only `DENY` is valid for ICMP actions). The command-line syntax for specifying an action is in the form: `ALLOW LOG_options SNMP_options` (for example, `ALLOW LOG_SUMMARY SNMP_NON`) or `DENY LOG_options SNMP_options ICMP_options` (for example, `DENY LOG_SUMMARY SNMP_NONE ICMP_HOST_UNREACHABLE`). The *first* underscore shown above with `LOG`, `SNMP`, and `ICMP` options is not required. For example, you can use either `LOG_SUMMARY` or `LOG SUMMARY`, `SNMP_NON` or `SNMP NON`.
- *Time* – (Optional) A component for setting up time-of-day-based Policy Rules. The command-line syntax for specifying time is in the form `TIME name_TIME`.
- *Screen* – (Optional) Screen name; command-line syntax is of the form `SCREEN name_SCREEN`.
- *Comment* – (Optional) Text that clarifies the purpose of a rule; command-line syntax is of the form `COMMENT "comment string."`

- *Encryption* – (Optional) The type of encryption (SKIP, IPSEC, or VPN) to be used when transferring packets. Encryption is only supported if ALLOW action is selected. It is not supported if any proxy or VPN is selected. Encryption works as follows:
 - If the first certificate is local (the secret key is loaded into SKIP or IKE on the Screen), the rule is considered an encryption rule. The packet will be encrypted on its way out of the Screen.
 - If the second certificate is local, the rule is considered a decryption rule, and the packet will be verified as having been decrypted accordingly (correct certificates and algorithms) before it is allowed to pass.
 - If neither certificate is local or if both certificates are local, the rule is ignored by the compilation process.
- *Proxy* – (Optional) The name of the proxy to be enabled and any flags for the proxy. Proxy flags are only permitted in a rule when the service is a PROXY_* service, where * is the type of proxy and the flags must be of the type for that proxy. They are only valid in a rule that has not specified any encryption parameters nor VPN. The command-line syntax for specifying a proxy is of the form: PROXY_*proxy_name proxy_flags*; for example, "PROXY_HTTP ACTIVE_X JAVA," "PROXY_FTP FTP_GET," or "PROXY_SMTP NO_RELAY." The Telnet proxy ("PROXY_Telnet") does not take flags.
- *User* – (Optional) A name in the proxy database of users. The command-line syntax for specifying a user is USER *user_name*, where *user_name* is a name in the proxy database.

Note – The optional USER field is *required* if the rule is PROXY_FTP or PROXY_Telnet, and is *optional* if the rule is PROXY_HTTP. Otherwise, the Screen does not support user-based rule criteria.

- *VPN* – A component that makes it possible to define your security policy with fewer rules because the Screen automatically generates the multiple rules containing all the SKIP or IKE information that the VPN defines. A VPN is group of Screens that transfer encrypted data among themselves over the public network. The command-line syntax for specifying VPN is of the form: *vpn_name*.

Note – VPN is not supported if any SKIP or IPsec information is specified, if DENY is selected, or if any Proxy is specified.

Example of a Rule Configuration

XYZ Company wants to set up SunScreen rules to implement the following security policy:

1. Allow telnet traffic from A (an individual host) to B (any host on a specified network).
2. Deny mail traffic between A and B; log attempts.
3. Deny all other telnet traffic and send NET_UNREACHABLE ICMP rejection messages for rejected traffic.
4. Discard all other packets.

The table below illustrates the SunScreen rules that the XYZ Company would set up to implement this security policy. Note that the default action for services not expressly mentioned in a rule would be specified as DENY.

TABLE 3-1 Sample Rules Table

Service	From	To	Rule Type	Log	SNMP	ICMP
telnet	A	B	Allow	NONE	NONE	NONE
mail	A	B	Deny	SUMMARY	NONE	NONE
mail	B	A	Deny	SUMMARY	NONE	NONE
telnet	*	*	Deny	NONE	NONE	NET_UNREACHABLE

Policy Versions

The *currently active policy* is the policy currently being used to filter network traffic. The currently active policy, which is displayed in the Policies List of the administration GUI and which can be determined from the command line by typing `ssadm active`, cannot be modified. You can make the common objects embedded in this version of the policy the current common objects, overwriting the existing set of common objects.

A regular policy (that is a policy without a version number) uses the current set of common objects and shares the common objects with other regular policies.

Each *version* of a policy has an associated version number. Edited policies automatically generate an historical version or snapshot of the common objects that are saved for reference. The version is shown by the incremental number after the dot in the name of the policy—the higher the number, the later the version. A version of a policy must be copied as a new policy with a new name before it can be activated.

Version numbering is implemented through the administration GUI or by using the `edit` subcommand of the `ssadm` command. A new version is created every time a configuration is saved.

A new version of a policy supersedes older versions. The older version still includes specific information and the actual content of the common object registry, not just a reference. An older policy version can become invalid because of changes to the network topology or to SunScreen host hardware or both.

The registry is the database of common objects. Each policy refers to a single registry. The versions of the policies have their own registry because the versions are histories.

You can copy a policy to a new name. You can edit a policy and save the changes to a new name to create a new policy. The rules can become the current rule for a policy; for example, the rules for policy Initial.10 can be made the rules for the current version of Initial.

Note – The rules created in this way are used with the current set of common objects. On verifying this policy, you may have to fix any inconsistencies.

Common Objects

Common objects are the components or data objects that you use in making up the rules for a security policy. Before you write these rules, you must add or define the common objects that you plan to use. This chapter describes the common objects that are used in configuring and administering SunScreen. It contains information on the following topics:

- “Service Object” on page 56
- “Address Object” on page 60
- “Screen Object” on page 64
- “Interface Object” on page 68
- “authuser Object” on page 73
- “Time Object” on page 75
- “proxyuser Object” on page 77
- “Jar Signature Object” on page 78
- “Jar Hash Object” on page 79
- “Certificate Object” on page 79
- “IPsec Key Object ” on page 81
- “Administration GUI Limitations” on page 81

After the common objects have been added to a policy, they are stored in a database and can be reused to create rule sets for additional policies. Common objects are of two types:

- Automatically-saved common objects that do not require saving after being defined or modified
- Common objects that require saving after being defined or modified

The following common objects are automatically saved when they are edited or new objects are added. You do not need to save these objects. Once these objects are added or edited, the change is saved immediately and is not repealed if the edit session is aborted. The Save button in the administration GUI remains greyed out, indicating that no Save is necessary due to such changes.

- Authorized User

- Jar Signature
- Jar Hash
- Proxy User
- logmacro
- mail_relay
- mail_spam
- vars

Note – Although the changes made to these objects are saved immediately, the changes do not take effect until a policy is activated.

All other common objects are not automatically saved after you define or modify them. The Save button in the GUI becomes active to show that they require saving because they are specific to the SunScreen's database. If you are using the configuration editor, you must save them with the `save` command.

All objects except Authorized User, Jar Signature, Jar Hash, Mail Relay, Mail Spam, Proxy User and the Screen object itself can have an optional Screen attribute. This attribute is normally specified by the syntax `SCREEN name_SCREEN`. The exceptions are for the `logmacro` and `vars` objects, which use the `sys=name_SCREEN` prefix.

Screen attributes designate that a particular object is to be used in favor of a non-specific object of the same name, when evaluated on the named `SCREEN`. It is possible and acceptable that there only be screen-specific objects for a given name (an underlying non-specific object need not be present, if it is never referenced). Nearly all pre-configured objects (except the Screen and Interface objects created by the initial configuration processes) lack Screen attributes.

Normally, Interface objects will have Screen attributes. Only when CMG will never be used, or in special situations, such as identical screens in an HA cluster, would warrant such.

Service Object

SunScreen is shipped with a number of predefined network services, such as `ftp`, `telnet`, `dns`, and `rsh`. See Appendix C for a complete list of services and service groups. These services describe the network protocol that is used in the packet-filtering rules. You manipulate them through the SunScreen administration GUI or the configuration editor. You can modify these services or define new services as needed.

Note – A well-defined service must not include two entries for the same port with different attributes, such as two different filters for the same port but with different state engines, or the same state engine but with different parameters set.

There are two types of service objects, described below:

- single service
- service group

Single Service

Part of setting up your network security policy is to define what network services will be available to hosts on your internal network and to hosts on the external network. Generally, for most sites you need to determine or set up rules that govern the basic services.

Besides the basic services, every TCP/IP implementation provides services such as `echo`, `discard`, `daytime`, `chargen`, and `time`. For services such as `ftp`, you can allow anyone in the internal corporate network to send outbound traffic, but only allow inbound traffic in this protocol to go to the public FTP server. This requires two rules: one for the outbound traffic and one for the inbound traffic going to the public server.

Each service uses a *state engine*, a sort of protocol checker. The state engine is a read-only object that describes the filtering behavior of the packet-filtering engines in SunScreen. For example, the FTP state engine checks port numbers when the `ftp` service is being used. For more information about state engines, see Appendix C. Table C-1 lists the predefined single service objects in SunScreen, along with the state engine and discriminator (port, RPC program number, or type). It also indicates the parameters (state engine modifiers, such as time-outs) and BROADCAST where applicable.

You can modify existing services or define new services as needed. Each service must use a predefined state engine. Troubleshooting is easier if you create a new service rather than modifying an existing one because you do not have to examine the services to see which ones you have modified. Note the purpose of the new (or edited) service in the description.

When you define a new service, you must specify a state engine for the new service to use and identify the various discriminators and parameters appropriate for that state engine.

You control the filtering activities by specifying what packet-filtering engine you want to use and the various discriminators and parameters applicable to that filtering engine.

Before you define a new network service, you need to identify how the new service will work:

- What protocol(s) does the new service use?
- What port(s) does the protocol use?
- If your service is an RPC protocol, what program numbers does it use?
- If your service is a UDP protocol, how many responses can be sent back for each request?

For example, if you have an FTP implementation that uses port 45 for its control port and port 44 for data, you could define a new FTP service called `ftp-45`. Refer to Appendix C for more information on state engines, their discriminators, and their parameters.

You can specify state engines as filters in both the forward and the reverse direction. The FORWARD filter applies when traffic originates from the From Address and goes to the To Address in a rule. The REVERSE filter applies to traffic originating from a machine in the To Address going to the From Address of a rule.

Normally, rules for stateful services do not have reverse filtering rules. For instance, an FTP connection is always established in the forward direction, and the returning traffic is handled by a state-table entry created when the connection is initiated. Reverse filtering rules are mostly valuable when you want to allow non-stateful traffic to return. An example is a rule that uses the `nlm` (network lock manager) service, which uses the non-stateful ICMP filter engine. It allows `nlm` requests (ICMP type 8) in the forward direction and `nlm` replies (ICMP type 0) in the reverse direction.

State engines can have a single port or a range of ports as a discriminator. A single port is just a number (#) ; a range of ports is denoted by a number-to-number range, written as #-# with no space before or after the hyphen. The keyword `PORT` or `BROADCAST` is used to identify the type of discriminator. When `BROADCAST` is specified for a service, the rules where the service is used allow communication to broadcast and multicast addresses. If you also want the service to work for non-broadcast addresses, you must include a filter discriminator both with broadcast selected (`BROADCAST`) and without it selected (`PORT`) .

Optionally, a set of parameters can immediately follow the port or range of ports that they modify. They are indicated by keywords and are a space-separated list of numbers.

A service's unique name is its given name plus the name of the Screen if it is associated with one. All references to a service are to its generic name, without regard to any associated Screen.

Service Group

A service group comprises a collection of single services and/or other groups of services. You can group network services together to apply a single rule to multiple network services. “SunScreen Network Service Groups” in Appendix C, shows the predefined service groups in SunScreen and the services each includes. Note that not every service is included in a service group.

The predefined common service group, for example, contains the following services: tcp, all, udp all, syslog, dns, rpc all, nfs, prog, icmp all, rip, ftp, rsh, real audio, pmap, udp all, pmap, and tcp all. You can create additional service groups using any combination of the individual network services. A useful group to define might be an “internet services group,” consisting of public services such as FTP, email, and WWW.

State engines that you use in describing services come in distinct *classes* and each class has *subclasses*. The subclasses form an order for choosing state engines when a rule includes a service group. Table 4-1 below shows state engines in order of preference—the greater the class/subclass number, the higher the preference. If you have a rule that uses a group of state engines, the one with the higher preference is matched.

A state engine that is followed by an asterisk(*) may conflict with another state engine because another state engine is in the same class and subclass.

TABLE 4-1 State Engine Class and Subclass

State Engine Name	Class	Subclass
nfsro	11	0
nis	10	0
pmap_nis	9	0
pmap_udp	8	0
pmtcp_tcp	7	0
realaudio*	4	1
rpc_tcp	6	0
rpc_udp	5	0
rsh*	4	1
sqlnet*	4	1
ftp*	4	1
tcpall	4	0

TABLE 4-1 State Engine Class and Subclass *(Continued)*

State Engine Name	Class	Subclass
dns*	3	2
ntp*	3	2
udp_stateless*	3	1
udp_datagram*	3	1
udp*	3	1
udpall	3	0
ping	2	1
icmp	2	0
ipmobile	1	3
iptunnel	1	2
ipfwd	1	1
ip	1	0
ether	0	0

A given service that you have defined manually to contain multiple state engines or in a service group that includes multiple services, can only contain a single state engine in a particular class or subclass for a particular port.

Address Object

An address object can be used in filtering rules, NAT rules, interface definitions, remote access rules, and VPN gateways. It can be an individual address, a range of addresses, or a group of addresses. You manipulate an address object through the SunScreen GUI or the configuration editor.

The addresses named `*` and `localhost` are reserved and cannot be modified in any way. The value of `*` is *all* IP addresses, as if a RANGE were defined with a starting IP address of 0.0.0.0 and an ending IP address of 255.255.255.255.

The value of `localhost` is the set of all addresses associated with the machine that SunScreen is running on—that is, the Screen's own IP address.

Optionally, you can associate addresses with a specific Screen object. In this case, its value is only used on the Screen with which it is associated. This approach enables multiple Screens to have different values for an address object with the same name. This is useful if, for example, you want to use the address “inside” and “outside” for all the Screens being managed.

An address object’s unique name is its given name plus the name of the Screen if it is associated with one. All references to addresses are to their generic names without regard to any associated Screen.

There are three types of address objects:

- Host Address
- Address Range
- Address Group

Host Address

SunScreen identifies a host address as an individual host by linking its unique IP address to an address object. This address object can use the name of the host, IP address of the host, or some other identifier. The figure below shows an example of a host identified by its IP address (172.16.1.1).

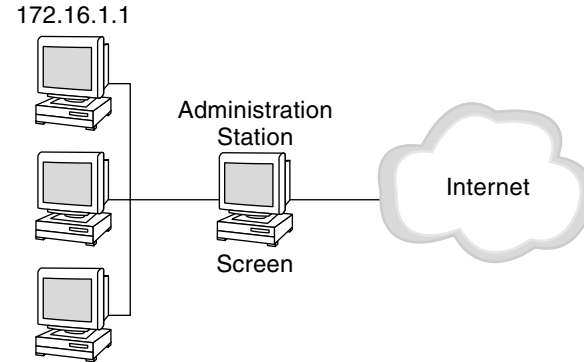


FIGURE 4-1 HOST – Single Address

Address Range

An address range is a set of numerically contiguous IP addresses. Networks and subnetworks are typically identified by a name for the range of IP addresses. You use the beginning and ending addresses to identify an IP address range.

You can set up an address object to represent an address range in SunScreen. The figure below shows examples of ranges of addresses for the Corporate and Sales networks. For example, the Corporate address object is defined as a range of addresses from 172.16.3.2 to 172.16.3.255. You could then establish access or encryption rules for hosts on the Corporate network by indicating the rule applies to the Corporate address object.

Note – SunScreen 3.2 supports four syntaxes for ranges:

- a.b.c.d e.f.g.h
- a.b.c.d - e.f.g.h (same as above, spaces optional around '-')
- a.b.c.d/m.n.o.p (m.n.o.p is the network+subnet mask)
- a.b.c.d/# (# is CIDR, number of network+subnet bits)

In the figure below, for example, the following are all the same:

- 172.16.3.0 172.16.3.255
- 172.16.3.0-172.16.3.255
- 172.16.3.0 - 172.16.3.255
- 172.16.3.0/255.255.255.0
- 172.16.3.0/24

The values in parentheses in the figure (255 . 255 . 255 . 0) represent a netmask.

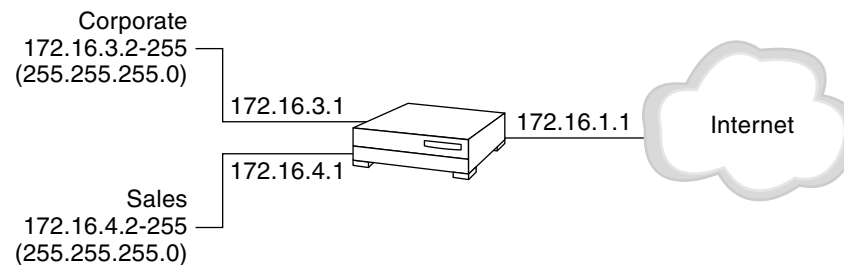


FIGURE 4-2 Range of IP Addresses

Address Group

An address group is a collection of host addresses, address ranges, and other address groups. After you set up an address group, you can use it to identify multiple hosts as a single element. You can define groups in terms of the addresses they include ("Address group A consists of the IP address 1.2.3.4 and the members of address group B"), the addresses they exclude ("Address group C consists of all the hosts on the 192.4.5.0 network except 192.4.5.5 and 192.4.5.9"), or both. Address groups cannot

be self-referential; that is, you cannot include address group X as a member of address group Y and then define address group Y as a member of address group X.

Note – There are two addresses you cannot modify: `localhost`, which is the IP address or addresses of the actual Screen and `*`, which represents *all* IP addresses.

The value of an address group is determined first by all included addresses, which means that all the IP addresses explicitly specified and all IP addresses contained in any other address groups included in the group are added to the address group. Next, all IP addresses of all the addresses and address groups that are excluded are removed from the address group. Note that you cannot control the order in which the IP addresses are added or removed: all includes are done before all excludes.

Designing an Addressing Scheme

You can take several steps when creating address objects to simplify maintenance of your security policies. When you are planning your addressing scheme, choose interface names that describe which addresses are on that interface or that reflect the names of the interfaces. Make naming conventions meaningful and consistent so that maintenance and daily administration are uneventful.

A network interface is a network connection coming into a Screen through which one or more IP addresses are accessible. These IP addresses need to be identified to SunScreen so that IP spoofing can be detected and prevented.

The easiest way to define address objects for network interfaces is to define an address group for each network interface. You can choose names that identify which addresses are on that network interface (such as, *Corporate*, *Sales*, *ftp-www*, and *Internet*) or names that identify the interfaces by type (such as `le0` or `qe0`).

In most cases, one interface has the majority of addresses on it. For example, the Internet network interface in the network illustrated in figure below has the most addresses, because it is the interface for all addresses except those in the Corp, ftp-www, and Sales networks.

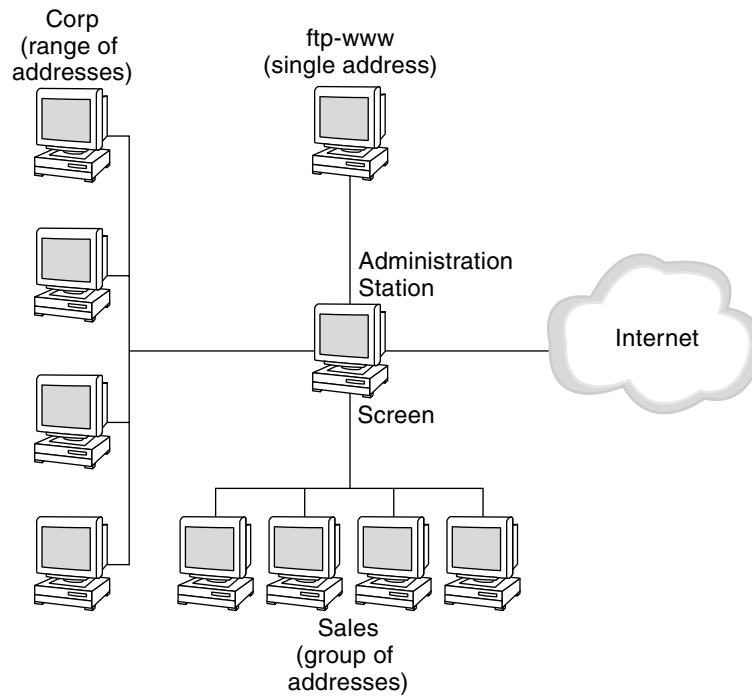


FIGURE 4-3 SunScreen as Internet Firewall

Rather than enumerating all the addresses for the Internet, you can define the address group for the Internet address object to include all network addresses (*) and then exclude those that you do not want to be part of that address. In the example shown in the figure, you would define the Internet address object as an address group that includes all addresses *except* Corp, Sales, and ftp-www. You would then define which hosts, networks, or address groups are members of the Corp, Sales, and ftp-www addresses to exclude them from the Internet address group.

Screen Object

A Screen object, which is created or modified through the configuration editor or the Administration GUI, maintains information about each Screen in the network and the relationships between the Screens.

The Screen object allows for an optional description of 256 characters or less in the comment field. It is also possible to edit:

- Miscellaneous Screen parameters
- SNMP information
- HA and CMG parameters
- Mail Proxy configuration

SunScreen Install automatically creates a local Screen object and gives it a name based on the output of `uname -n`. Additional Screen objects must be created when setting up a High Availability (HA) cluster or a Centralized Management Group (CMG).

Note – The Screen named `*` is created by default and cannot be modified in any way. This name is reserved and is used to indicate *any* Screen when an object or rule requires a Screen name to be specified.

Miscellaneous Parameters

There are various flags that can be set within a Screen object when you use the configuration editor. These flags indicate:

- Whether or not Destination Address Checking is enabled (`DEST_CHECK`)
- If the Screen will allow or deny routing traffic (`RIP`)
- Which naming services it will use (`DNS`, `NIS` or `none`)
- Whether the Screen uses SKIP certificate discovery protocol (`CDP`)

Destination address checking checks the destination IP address of each packet coming out of a Screen. The packet is allowed to go out only if its destination IP address is defined on the interface. Those addresses are defined by listing them in the valid addresses for an interface. See “Interface Object” on page 68 for a discussion of valid addresses. Destination address checking protects against sending packets out on the wrong interface.

Note – Destination address checking is an attribute of the Screen object. You enable or disable it Screen-wide. You can’t enable it on only some interfaces.

The size of the log file (`LOGSIZE`) is an optional parameter which is specified in megabytes. If no size is specified, the default size is 100 MB.

If the Screen is configured in stealth mode, the network that it partitions and the netmask must be specified. In the configuration editor this is accomplished using the `STEALTH_NET #.#.#.#.#.#.#` keyword, where the first `#.#.#.#` is the network address and the second `#.#.#.#` is the netmask. In the administration GUI, these parameters are the *Stealth Net Address* and *Stealth Netmask*, respectively, in the Miscellaneous tab of the Screen object.

SNMP Information

In order to issue SNMP alerts, a list of SNMP receivers and their respective IP addresses must be specified. The SNMP time status indicator can also be enabled by setting the SNMP timer interval in minutes. These parameters are set in the configuration editor with the `SNMP` and `SNMP_TIMER` keywords, or in the administration GUI under the SNMP tab of the Screen object.

The following SNMP traps are supported:

- An action on a packet that matches a particular rule
- A default drop action on an interface
- Time status indicator traps

The first two types include the following data:

- `interface` – The SunScreen network interface number on which the packet was received.
- `interfaceName` – The SunScreen network interface name on which the packet was received.
- `errorReason` – The reason the alert was generated. (See the `sunscreen.mib` file for a complete list of reasons.)
- `packetLength` – The actual length of the packet in bytes.
- `lengthLogged` – The length of the data logged in bytes.
- `packetData` – The packet data.

The SNMP timed status indicator trap uses the same receivers database as other types of SNMP traps. There is only one database with a maximum of five receivers. These receivers are specified as variable to the screen object.

The following data are in the SNMP timed status indicator. These data cannot be modified and new data cannot be added:

- `cpuUsage` – Average percentile CPU usage
- `memoryAvail` – Current swap space available, in kilobytes
- `swapIn` – Current swap ins
- `swapOut` – Current swap outs
- `scanRate` – Current scan rate
- `tcpUsage` – Current number TCP connections in the SunScreen state table
- `ipUsage` – Current number IP connections in the SunScreen state table
- `udpUsage` – Current number UDP connections in the SunScreen state table
- `rootUsage` – Disk usage of the root partition, /
- `varUsage` – Disk usage of the var partition, /var
- `etcUsage` – Disk usage of the etc partition, /etc

- tmpUsage – Disk usage at the tmp partition, /tmp

Only these SNMP traps are supported. No get or set operations are supported.

If you want the Screen to use SNMP time status indicator, you must set the SNMP_TIMER keyword with a time value in minutes. You must have defined the SNMP receivers to use this feature. If it is not set, it is not enabled.

Note – SunScreen 3.2 supports four syntaxes for ranges:

- STEALTH_NET a.b.c.d e.f.g.h
 - STEALTH_NET a.b.c.d - e.f.g.h (same as above, spaces optional around '-')
 - STEALTH_NET a.b.c.d/m.n.o.p (m.n.o.p is the network+subnet mask)
 - STEALTH_NET a.b.c.d/# (# is CIDR, number of network+subnet bits)
-

HA and CMG Parameters

When a Screen is part of a High Availability (HA) cluster or Centralized Management Group (CMG), it is classified as either a Primary or Secondary Screen. Primary and Secondary are defined as follows:

- Primary - One of the Screens in the collection must be the primary Screen, and all other Screens are Secondary. The primary Screen is used for all administration: it holds the configuration data and is responsible for communicating with the Secondary Screens in order to update their policies.
- Secondary - A Secondary Screen receives all its policy information from the primary Screen. A Secondary Screen must indicate the name of the primary Screen in its Screen object. This name is specified with the MASTER keyword in the configuration editor or the Primary Name field under the Primary/Secondary tab in the administrative GUI.

Note – The primary Screen object has no Primary name specified, but is recognized as the primary Screen if its name appears in the Primary Name field of at least one other Screen.

If the Screen is part of a centralized management group or is administered remotely, you must specify the following field:

- ADMIN_IP – An IP address that can be used to communicate with the Screen for administrative traffic. You specify it as an IP address in the form #.#.#.#.

If SKIP is used to protect these data communications, you must specify the following fields:

When configuring interface objects, you must specify the type of interface being used. SunScreen uses five types of interfaces, which are discussed later in this section. Note that when using the configuration editor to add or modify an interface, the interface type must be typed using all caps as shown below:

- ROUTING
- STEALTH
- ADMIN
- HA
- DISABLED

Note – See the *SunScreen 3.2 Installation Guide* for detailed information about interface types and their installation, including a caution about mixing routing and stealth interface modes. See also the *SunScreen 3.2 Administration Guide* and the *SunScreen 3.2 Configuration Examples*, the latter of which includes an example of a mixed-mode configuration.

You can, as an option, associate any interface with a specific Screen object. If the specified Screen name is part of a centralized management group (CMG), this association is necessary to distinguish which interface definition belongs to which Screen.

In addition to the interface type and Screen parameters, there are several other parameters which are part of the interface object. These parameters are defined as follows:

- **Description** – The interface description allows for an optional COMMENT of 256 characters or less.
- **Valid Addresses** – Specifies the set of addresses reachable through a particular interface. The interface's valid address set is the basis for packet forwarding decisions in stealth mode and for source address spoof protection in routing mode.
- **Spoof Protection** – Each routing interface is marked as having a complete or incomplete valid address set.

If the valid address set is **complete**, only the addresses defined in the set are allowed as the source address for packets received on the interface. Any other addresses are considered "spoofed" and the packets are dropped.

If the valid address set is **incomplete**, some of the source address spoof protection is disabled. Specifically, addresses defined in another interface's valid address set are disallowed on the incomplete interface, but addresses that are *not* explicitly assigned to another interface are considered legal, and therefore allowed.

Note – If the valid address sets are empty and `incomplete` is set (default after routing mode install), *all* source addresses are valid on *all* interfaces.

If the address sets are empty, and `complete` is set, *no* source addresses are valid on *any* interface. The Screen will not pass any traffic if configured like this.

Spoof checking is the first thing that is done to a packet when it arrives on the interface. No other SunScreen processing is performed unless the packet passes the spoof check.

- **Address Overlap** – Each routing interface optionally may specify an overlap parameter that contains a set of addresses that are allowed to overlap between this interface's valid address set and that of another routing interface on the same Screen. Otherwise, it is an error to define two routing interfaces that have overlapping valid address sets. This option is not available for stealth interfaces.
- **Logging** – Identifies the disposition of a packet when a packet arriving on an interface does not match any rule or does not pass the spoof check. It has the values:
 - **None** – Do not log.
 - **Summary** – Record the first 40 bytes of the packet in the log.
 - **Detail** – Record the complete packet in the log.
- **SNMP Alerts** – Specifies whether the Screen should issue an SNMP alert when a packet received on an interface does not match a rule or does not pass the spoof check. the options are:
 - **SNMP_NONE** – Do not send an SNMP alert message. This is the default.
 - **SNMP** – Send an SNMP alert message to the SNMP receivers listed in the Screen object when a packet received on this interface is rejected.
- **ICMP Action** – Identifies the ICMP rejection message that is issued if a packet received on the interface is rejected. In most cases, the Screen rejects packets by sending and ICMP Destination Unreachable packet with the reject code set as specified in the ICMP action on the interface. However, if a TCP packet matches a DENY rule, and the ICMP action is set to `PORT_UNREACHABLE`, a TCP RESET packet is issued instead. It has the options:
 - **NONE**
 - **NET_UNREACHABLE**
 - **HOST_UNREACHABLE**
 - **PORT_UNREACHABLE**
 - **NET_FORBIDDEN**
 - **HOST_FORBIDDEN**
- **Router IP Address** – Specifies the router's address when the Interface type is stealth. A stealth interface may define up to five routers, by IP Address. These routers are used for forwarded traffic not on the local network.

Routing Interface

Routing interfaces have one or more IP addresses and route packets using the standard IP routing mechanisms in the operating system. Each routing interface is connected to a different IP network just like a standard IP router. In terms of network placement, a Screen with routing interfaces replaces a router. A routing interface can also receive administrative traffic from a remote Administration Station.

Connections to and from proxies can only occur through ROUTING interfaces. Thus, to run proxies or if you want to install additional network services on the Screen, you must configure the Screen to have at least one routing interface.

Use routing interfaces to:

- Replace an existing router
- Control packet flow between different IP networks
- Receive administrative traffic for the SunScreen configuration
- Install proxies or other network services on the Screen

A Screen with routing interfaces does not need a separate admin interface, because the administrative traffic can come in through one of the routing interfaces. If Screens are part of an HA cluster, they each must have a unique HA interface for the dedicated HA heartbeat network.

Stealth Interface

Stealth interfaces have no IP address. A Screen with stealth interfaces partitions an IP network and controls packet flow between the partitions. Screens containing only STEALTH interfaces are required to have one ADMIN interface for administrative traffic.

Note – Although it acts much like an IP bridge or switch, a Screen with stealth interfaces does not implement the bridging algorithms that detect loops. Make sure that no loops exist in your network configuration where a packet could be sent out from one stealth interface and be received on another. Also note that HA (high availability) clusters require that the machines be connected by means of a non-switching hub.

Stealth interfaces provide a higher degree of security than routing interfaces because they are separate from the standard IP mechanisms used by the operating system. Thus, packets flowing through stealth interfaces cannot inadvertently leak into other network applications running on the system, thereby compromising the security of the firewall.

Admin Interface

An admin interface is a special case of a routing interface configured only to pass administration traffic for the Screen. An admin interface is not required for a Screen with routing interfaces because routing interfaces can pass administration traffic through to the Screen.

Note – Because stealth interfaces have no IP address, they cannot provide the IP address needed for administration traffic. You must, therefore, configure a Screen that has only stealth interfaces with an additional admin interface.

Routing and Stealth Interfaces on a Single Screen

You can configure a Screen with a mixture of routing and stealth interfaces subject to the following restrictions:

- Packets do not flow between the routing and stealth interfaces. Packets received on a stealth interface are only sent out on another stealth interface. Packets received on a routing interface are only sent out on another routing interface.
- Any packet affected by NAT or encryption must only pass through the Screen once.

HA Interface

Each Screen that is part of an HA cluster must have a single, unique HA interface for the dedicated HA heartbeat network. It is possible to administer the HA cluster over this interface, but it is primarily for synchronizing the Screens within the cluster and passing configuration data from the primary Screen to the secondary Screens.

Disabled Interface

An interface of type DISABLED does not filter any traffic. It is important to understand that traffic can still flow across such an interface if it is configured "up" within Solaris. Care should be taken to understand the possible ramifications of using a DISABLED interface in this manner.

If the Screen contains ROUTING interfaces, it is possible for packets to flow between the DISABLED interface and the ROUTING interface (due to Solaris routing). The packets entering or leaving the disabled interface are not filtered, but the packets leaving the Screen over the ROUTING interface are filtered.

If the DISABLED interface is defined on a Stealth-mode-only Screen, it will pass no traffic.

authuser Object

The `authuser` (authenticated user) common object specifies the means by which users are to be validated for their roles in `proxyuser` objects. SunScreen administrative privileges are also often granted based on `authuser` objects. The `authuser` common object is automatically saved when it is edited or a new authorized user object is added. The change is not repealed if the edit session is aborted. The Save button in the administration GUI remains greyed out, indicating that no Save is necessary due to such changes.

An `authuser` object is one of the data objects that defines a SunScreen configuration. You manipulate it through the configuration editor or administration GUI. `authuser` objects are the repositories for authentication and demographic information used to identify individual users. This information is used to authenticate users by a Screen's access control facilities.

Each `authuser` object has a name. The name cannot contain the following characters:

! # \$ % ^ & * { } [] < > " \ ? ' / @ NULL

You can choose the names to coincide with existing, real-world naming schemes for individuals. Thus, "harold.bovis", "Sally Ann Studebaker", and "Rundum, Karr Bo" are examples of legitimate `authuser` names. The name space of the authorized user is separate from all others in the SunScreen firewall. (In particular, `authuser` names are different from those that name `proxyuser` objects.)

Tip: Unlike the names of `proxyuser` objects, the names of `authuser` objects are rarely entered by the user directly. The exception to this is their (optional) direct use in administrative access rules (`accesslocal` and `accessremote`).

`authuser` objects store information describing individual users of interest to various SunScreen access control facilities. The data contained within its entries fall into three general groups:

- authentication
- demographics
- control

Authentication information is employed by the processing in the SunScreen firewall to confirm the identity of a potential user. Three types of authentication are supported: simple password, SecurID, and RADIUS. RADIUS authentication does not use

authuser objects. A given user object can specify the use of either or both of the other two types simultaneously. Authentication processing attempts to match any password or passcode entered against each type specified in the order present within the entry's record.

(The preceding statement is true within certain limits. For example, a password that cannot possibly be a SecurID passcode will never be presented to that mechanism even if SecurID is specified. If you use the SecurID type, it should be given after all other types.)

Note – The Java-based graphical configuration tools only allow for a single, simple password type or a SecurID type, in that order. Both the authuser objects and the SunScreen authentication processing allow multiple, simple password types to be specified, and each will be tried in the order present. However, entries with multiple, simple password types will not be properly displayed or edited by the Java-based tools.)

Demographic information stored in authuser objects is used to identify users better and to improve and possibly automate user contact:

- `DESCRIPTION="desc"` – Specifies a plain-text description string `desc` to be associated with the user entry.
- `REAL_NAME="namestr"` – Specifies an optional real name string to be associated with the authuser object.
- `CONTACT_INFO="contactstr"` – Specifies an optional contact information string to be associated with the authuser object (for example, email address).

Control items are used by the authentication logic to restrict processing, and the like. Each authentication item can have an individual enablement tag, which determines if that particular item is to be processed. The entire object also has such an enablement tag, allowing a user's entry to be turned-off without deleting it. (Technically, the structure that stores the name is also a control item.)

- `ENABLED` – Allows authentication processing to consider this object. This is the default.
- `DISABLED` – Does not allow authentication processing from considering this object.
- `PASSWORD={ "pwd" pwdarg . . . }` – Specifies a simple password method to be used for authentication. The `pwd` field is the user password, in plain-text. The processing for the `PASSWORD` keyword automatically translates this string from plain-text into crypto-text, and causes a `CRYPT_PASSWORD` subitem to represent `pwd`. Thereafter, the plain-text password is displayed as an empty string, with the `CRYPT_PASSWORD` subitem retaining the password. The subitems for the `pwdarg` are:
 - `CRYPT_PASSWORD="cryptpwd"` – The DES-encrypted version of the user's password. This value retains the password in a form that avoids its

compromise. When presenting this value to the add verb, the value of pwd should be empty (""). (The password encryption mechanism is the same one used by Solaris' native user password scheme.)

- **ENABLED** – Allows authentication processing to attempt to match this password item. This is the default.
- **DISABLED** – Does not allow authentication to attempt to match this password item.
- **SECURID={ "sidname" sidargs ... }** – Specifies the SecurID method to be used for authentication. The sidname is the string in the login: or Default login: field in the SecurID server entry for this user. The subitems within sidargs are:
 - **ENABLED** – Allows authentication processing to attempt to match this SecurID item. This the default.
 - **DISABLED** – Does not allow authentication processing to attempt to match this SecurID item.

Time Object

Time objects are specified using a 24-hour clock in 5-minute increments. You use time objects to set a policy's time-of-day, day-of-week, and the like in *time-based rules*. For instance, you can allow telnet, but only during regular business hours, or after hours, or outside certain hours.

The policy rule default setting is ANY time, which applies the rule at all times. You can set a few time-based policy rules that reference the same set of hours, or you can specify the hours covered for a particular day or range of days, such as Monday to Friday, 9 a.m. to 5 p.m., local time.

Time is always interpreted as the Screen's time zone, which requires that you either have Screen-specific time definitions to coordinate traffic between the Screens in different time zones, or have distinctly named time objects and Screen-specific rules.

Note – Although you can define many time objects, only 31 distinct time objects can be in actual use on any given Screen. You cannot modify the time object named * in any way; it represents 24 hours a day, 7 day a week. It is the same as if no time object is used. It is not included in the limit of 31 time objects.

For example, Los Angeles (LA) and New York (NY) have a three-hour difference. Suppose each site is protected by a Screen. If you only want the two sites to communicate when they are both within "regular" hours (that is, 8 a.m. to 5 p.m.),

then NY is available to communicate to LA between 11 am and 5 p.m., and LA is available to communicate to NY between 8 a.m. and 2 p.m.

A downside to this is that during hours that do not overlap, one of the two Screens allows traffic through while the other does not. So, early in the morning the NY Screen allows traffic through, but it is blocked by the LA Screen. Similarly, in the afternoon, the LA Screen is blocked by the NY Screen.

Case 1: Screen-Specific Time Objects

Name	Screen	Value
regular	NY	MONDAY { 11:00 17:00 } TUESDAY { 11:00 17:00 } ...
regular	LA	MONDAY { 08:00 14:00 } TUESDAY { 08:00 14:00 } ...

Set the rules by typing:

```
telnet LA NY TIME regular ALLOW
telnet NY LA TIME regular ALLOW
```

These rules apply to both Screen, although the definition of regular is different for each Screen

Case 2: Distinctly Named Time Objects

Name	Value
ny-business	MONDAY { 11:00 17:00 } TUESDAY { 11:00 17:00 } ...
la-business	MONDAY { 08:00 14:00 } TUESDAY { 08:00 14:00 } ...

Set the rules by typing:

```
SCREEN LA telnet LA NY TIME la-business ALLOW
SCREEN LA telnet NY LA TIME la-business ALLOW
SCREEN NY telnet LA NY TIME ny-business ALLOW
SCREEN NY telnet NY LA TIME ny-business ALLOW
```

proxyuser Object

The proxyuser common object contains the mapping information for users of SunScreen proxies. You manipulate the proxyuser through the administration GUI or the configuration editor. The proxy user object has the subtypes single and group. FTP, Telnet, and (optionally) HTTP proxy rules reference the proxy user entries.

The proxyuser object is automatically saved when it is edited or a new proxyuser object is added. The change is saved immediately and is not repealed if the edit session is aborted. The Save button in the administration GUI remains greyed out, indicating that no Save is necessary due to such changes.

proxyuser objects store associations between authuser objects and a user ID (or other host-based user identifier) to be used by a proxy to authenticate and establish a user's identity on a "backend" server system. These associations, or "mappings", enable reusing the authentication information within authuser objects; by creating multiple mappings, any given authuser can be cast into different roles with respect to one's identity on different "backend" servers to be proxied. These mapping proxyuser objects are dubbed "simple" ones.

Finally, some proxyuser objects are considered "special" ones. Such objects are similar to simple objects but provide access paths to one or more user entities that can be authenticated by an external mechanism. RADIUS and SecurID are two such external mechanisms that are presently supported.

proxyuser objects are referenced in SunScreen proxy policy rules.

Each proxyuser object has a name. The name cannot contain the following characters:

`! # $ % ^ & * { } [] < > " \ ? ' / @ NULL`

You can choose names that coincide with existing real-world naming schemes for individuals, existing computer-based naming, or any other scheme. Thus, "hbovis (mechengg)", "sally.ann.studebaker", and "Rundum, Karr Bo – security" are all examples of legitimate proxyuser object names. The namespace of the proxyuser objects is disjoint from all others in the SunScreen firewall. (In particular, proxyuser names are different from those that name authuser objects.)

Tip: Choose names for proxyuser objects which can be readily entered by users.

Each proxyuser object has an enabled tag, so that you can turn it off for processing purposes without deleting it:

- **ENABLED** – Allows authentication processing to consider this object; this is the default.

- **DISABLED** – Disallows authentication processing from considering this object.

The two portions of the simple object mapping are: an authuser object reference and backend user name. Each of these is an optional field:

- **AUTH_USER_NAME="authusername"** – Specifies a reference to an authuser object with the given authorized user name. If this item is not present, then the proxyuser object does not require authentication.
- **BACKEND_USER_NAME="backendname"** – Specifies the system-dependent name, to be supplied to the backend server as the identity of the user of this mapping. (At present, this item should always be defined.)

In addition to simple entries, proxyuser objects also enable creating groups of sibling objects. The GROUP is a way to group proxy users that have the same privileges. Group proxyusers save time when creating rules. Before creating a proxy user group, define the proxy user objects for that proxy user group. These GROUP objects can contain zero or more references to other simple or group proxyuser objects. The group structures are maintained hierarchically, not flattened. For GROUP proxyuser objects the value fields are:

- **MEMBER_NAME="mname"** – Makes a reference to another proxyuser object mname by the one containing it. This item can be repeated as many times as needed to make references to all group members.

Any proxyuser object, regardless of type, may have the following optional attributes:

- **DESCRIPTION="desc"** – Specifies a plain-text description string desc to be associated with the user entry.

Jar Signature Object

The Jar signature common object identifies the Java archives (JARs) that you want the Screen to pass. You can manipulate it through the administration GUI or the configuration editor. JAR signatures apply only to the HTTP proxy.

The Jar signature object is automatically saved when it is edited or a new Jar signature object is added. The change is saved immediately and is not repealed if the edit session is aborted. The Save button in the administration GUI remains greyed out, indicating that no Save is necessary due to such changes.

Jar signature objects are named, and their name space is distinct from all others within SunScreen. The name of a Jar signature object is entirely ephemeral; it is not referenced in any other context by SunScreen. The name consists of 1 to 255 characters, not including the following:

!#\$%^&*{}[]<>“\?’ / @ NULL

Each Jar signature object has a value part. This value is a 32-character hex digit MD5 hash value. It is used to verify the integrity of Java archives (JARs) that are allowed through the SunScreen HTTP proxy.

Jar Hash Object

The HTTP proxy can be set up to filter the Java applets based on the hash value of the Jar file. Each Jar hash object has a value part. This value is a 32-character hex MD5 hash value. It is used to verify the integrity of Java archives (JARs) which are allowed through the SunScreen HTTP proxy.

A Jar hash object is one of the data objects that define a SunScreen configuration. You can manipulate it through the configuration editor or the administration GUI. A Jar hash object is automatically saved when it is edited or a new Jar hash object is added. The change is saved immediately and is not repealed if the edit session is aborted. The Save button in the administration GUI remains greyed out, indicating that no Save is necessary due to such changes.

Jar hash objects are named, and their name space is distinct from all others within SunScreen. The name of a Jar hash object is entirely ephemeral. It is not referenced in any other context by SunScreen. The name consists of 1 to 255 characters, not including the following characters:

!#\$%^&*{}[]<>“\?’ / @ NULL

Certificate Object

A certificate object is a mapping between a name that users can read and a SKIP or IKE identity. You manipulate certificate objects through the administration GUI or the configuration editor and use certificate objects to configure the certificates that you use for secure communication between your Screen and remote hosts. Remote administration is only possible using certificate objects.

Note – Changes to the certificate object that pertain to loading into SKIP or IKE take effect immediately without having to be saved. On the other hand, changes to the certificate object as stored in the common objects registry do *not* take effect immediately and must be saved explicitly. They only take effect when the policy in which they are used is activated. For example, if you add a new certificate, the certificate is created and loaded immediately into SKIP or IKE, but the name has *not* been saved as part of the common objects, and must be saved explicitly. Similarly, if you rename a certificate, you must explicitly save it.

You can write an optional description for all certificates in the comment field. The description is limited to 256 characters or less.

Optionally, you can associate a certificate object with a specific Screen. If you associate a certificate object with a Screen, its value is used only on that Screen.

A certificate's unique name is its given name plus the name of the Screen, if any, with which it is associated.

There are two subtypes of certificates:

- Single Certificate
- Certificate Group

Single Certificate

A single certificate object represents a single SKIP or IKE identity. A SKIP certificate object has an NSID (name space identifier) and an MKID (master key identifier). An IKE certificate object is identified by the Subject Name DN (Distinguished Name).

You can assign a name to a SKIP or IKE certificate that already exists. The certificate object provides a way to associate a usable name with a SKIP certificate NSID/MKID pair or an IKE DN. This naming facility makes using certificates easier, as well as isolating the Screen configuration from exact SKIP or IKE names. You associate a certificate ID when you want to encrypt communication between two Screens or between a Screen and a remote Administration Station.

Certificate Group

A certificate group allows grouping single certificates that you want to use together. In the Administration GUI select New Group... from the Add New Object pulldown to create a certificate group.

IPsec Key Object

An IPsec key object is a mapping between a human-readable name and a hexadecimal string of keying material to be used for manually-keyed IPsec communication or IKE authentication using a pre-shared key. It is manipulated with the configuration editor or the administration GUI.

Administration GUI Limitations

The administration GUI performs almost all the normal administration tasks, but it does not support every option of the configuration editor for use with the command line interface. The configuration editor offers many options for each command. Appendix B contains information about the configuration editor.

Administration

This chapter describes the concept of Screen administration and possible ways to administer a single Screen or multiple Screens. It contains information on:

- “Administering the Screen” on page 83
- “Local Administration” on page 84
- “Remote Administration” on page 84
- “Centralized Management Group” on page 85
- “Logging” on page 86
- “Common Objects Used in Centralized Administration” on page 86
- “Creating Common Objects and Policies for Multiple Screens” on page 86

Administering the Screen

SunScreen has two types of components for administration: a *Screen* and an *Administration Station*. The two components can be installed separately and remotely or they can be installed locally on a single system.

You can administer Screens either through the administration GUI or through the command line. See the *SunScreen 3.2 Administration Guide* for GUI procedures and Appendix B for information about the command line.

You typically choose whether to administer a Screen locally or remotely when you create the initial SunScreen configuration. You can also add a remote Administration Station after the SunScreen software has been installed. A system that is being administered remotely can be *headless* (no monitor) and have no keyboard.

The number of Screens and Administration Stations needed at a site depends on its network topology and security policies. Typically, one Screen is installed at each

network direct public access location that needs to be restricted. One or more Administration Stations can manage multiple Screens.

Object definitions, policies, logs, etc. reside on the Screen(s). logmacros and the logs themselves reside on all Screens (master, slave, primary, secondary). All other configuration information resides only on the master Screen in a CMG. Multiple admin stations have equal and uniform access to the same configuration information. This provides for multiple administrators, mobile administrations, and other flexibility advantages. Once logs are downloaded (via the log browser and/or the command line log manipulation tools), their accessibility is outside of the provisions of the SunScreen management model. The administrative organization, access control, policies and procedures, and log retrieval, analysis, and archival should be carefully planned and articulated to users.

Local Administration

Local administration is performed on the same host where the Screen software is installed, as shown in the figure below. Because administrative commands do not travel over a network, local administration does not require encrypted communication.

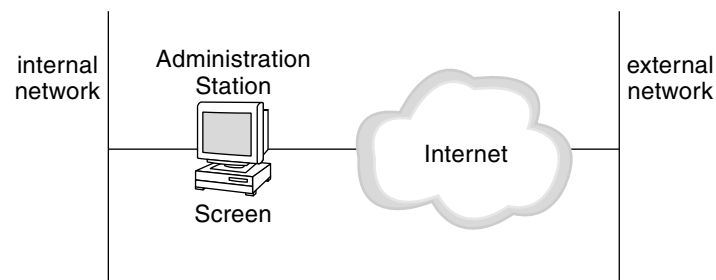


FIGURE 5-1 Local Administration of a Screen in Routing Mode

Remote Administration

For *remote administration* of a Screen from an Administration Station, install the software packages, including SunScreen SKIP and/or IKE, on separate systems, as shown in the figure below. In the figure, a remote Administration Station on the internal network administers the Screen located between the internal network and the Internet. This Screen is the router between the internal network and the Internet. A second remote Administration Station for this Screen is located on the external

network. Note that communication between a remote Administration Station and a Screen must be encrypted.

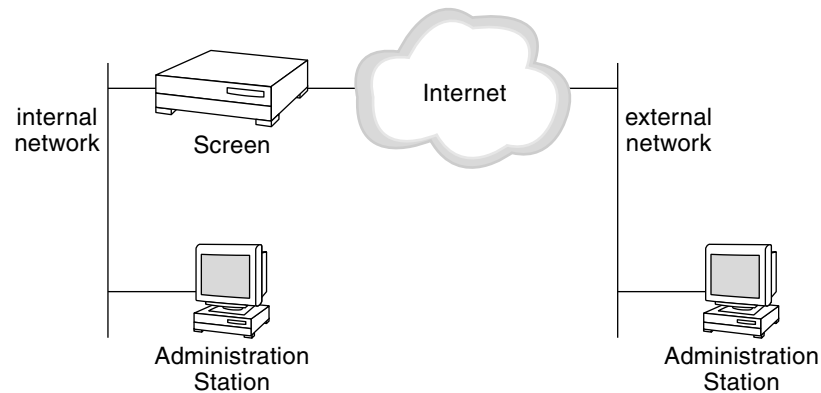


FIGURE 5-2 Remote Administration From an Administration Station to a Screen in Routing Mode

Centralized Management Group

A centralized management group (CMG) is a group of Screens that can be deployed throughout your organization. A CMG is managed with a set of configuration objects through an Administration Station or several Administration Stations.

- **Primary Screen** – Policies and all configuration objects reside on a specific Screen called the CMG's primary Screen. The centralized management group's primary Screen manages itself, as well as the centralized management group's secondary Screens. It is configured with all the common objects and rules for all the Screens in the centralized management group. The primary Screen can be administered locally or from a remote Administration Station, or both. Each primary Screen can nominally support up to 20 secondary Screens, although there is no hard limit. It is limited only by performance.
- **Secondary Screens** – The default configuration on the secondary is only the initial setup policy that was created during installation of the software so that the primary Screen could communicate with it. You cannot directly edit a policy on a secondary Screen, once that Screen has joined a CMG.

Logging

The logs are stored in a circular buffer on each Screen. You can download them to the Administration Station for analysis, archival, etc. Although no central logging mechanism exists for a global view of the logs on the individual Screens in a centralized management group, the secondary Screens make some basic emergency administration possible. For example, if the primary Screen is down for service, you can select a specific Screen and view its log.

Note – If you have configured your centralized management group with multiple Screens and the Screens' names differ from the host name, be sure to use the correct host name if you want to view the information for that Screen. This name must match the host name in the `/etc/hosts` file.

Common Objects Used in Centralized Administration

Centralized administration requires secure communication among the Screens. This information is contained in the screen object. On the primary Screen, screen objects must exist for all the Screens. On each secondary Screen, Screen objects must exist for that secondary and the primary Screen.

Note – Once you successfully activate a configuration from the primary Screen, it will replace objects on the secondary. If these new objects are incorrect, you may not be able to activate additional configurations centrally. If so, you can manually activate an old configuration on the secondary, fix the errors on the primary, and then activate the configuration again.

Creating Common Objects and Policies for Multiple Screens

When creating common objects and policies for multiple Screens, the object or policy rule by default applies to all Screens controlled by that primary Screen. You can restrict an object or rule to a single Screen by specifying its name in the Screen field in objects and rules.

While you could restrict all your objects and rules to a single Screen, the power of centralized administration comes when you can use common objects and rules to apply to multiple Screens.

Most common objects are applicable to all Screens, but sometimes an object such as an address object called `Inside` may be different on different Screens. In this case, make the names unique by adding a suffix or prefix to the name (for example `Inside-East` and `Inside-West`) rather than using the `Screen` option to restrict the scope of the object.

Interface Objects

You generally need to limit interface objects to a specific Screen because the names must be the name of the network interface on that system. Because the names of these objects must match those of the Solaris network devices they configure, use the `Screen` entry in the interface object to restrict that object to a single Screen.

Policies

Policies for a centralized management group are the same as any other policies and consist of a set of rules that control the behavior of the centralized management group. You set up rules for the entire centralized management group of Screens using the administration GUI.

Policies and all configuration objects reside on the primary Screen. The primary Screen pushes the rules that apply to the remote Screens out to the Secondary Screens when the policy is activated. Each rule in the policy can be applied to all Screens or just one.

The policy is sent over an encrypted connection to the secondary Screens. The policy is then compiled locally on each secondary Screen. The compiled policy is stored on the primary Screen.

The primary Screen “pings” the secondary Screens before it activates a policy and sends the administrator a message if there is a problem. The primary can push a policy to the other secondary Screens in a centralized management group even if one of the secondary Screens doesn’t respond to the ping. A policy that is being pushed out to the secondary Screens is activated in parallel on all the secondary Screens. The primary Screen does not have to wait for each secondary Screen to compile the policy separately before sending it out to the next secondary Screen.

You cannot edit an activated policy on the primary Screen. You also cannot directly edit a policy from a Secondary Screen.

Encryption, Tunneling, and Virtual Private Networks

Encryption is the process by which a readable message is converted to an unreadable form to prevent unauthorized parties from reading it. *Decryption* is the process of converting an encrypted message back to its original (readable) format. This chapter includes discussions of the following:

- “Encryption and Decryption” on page 89
- “How SunScreen Uses Encryption” on page 90
- “Packet Examination” on page 90
- “Tunneling” on page 91
- “Using IKE With SunScreen” on page 93
- “IKE Options” on page 94
- “IKE Policy Rules” on page 96
- “Defining a VPN” on page 98
- “Adding a VPN Rule” on page 101
- “VPN Limitations” on page 103

Encryption and Decryption

An unencrypted message is called a *plaintext* message. An encrypted message is called a *ciphertext* message.

Digital encryption algorithms work by manipulating the content of a plaintext message mathematically, using an encryption algorithm and a digital key to produce a ciphertext version of the message. The sender and recipient can communicate securely if the sender and recipient are the only ones who know the key.

Encryption is important to SunScreen because it provides a mechanism for protecting the privacy of communications and authenticating the identities of the sender and receiver. Encryption technology enables you to authenticate systems and users. As a

result, you can define rules that control access according to specific cryptographic identities rather than according to general IP addresses.

SunScreen uses either IPsec/IKE (Internet Protocol Security Architecture/Internet Key Exchange) or SKIP (SunScreen Simple Key Management for Internet Protocols) as the basis for its encryption technology. Both IKE and SKIP provide secure, encrypted communication between a remote Administration Station and a Screen and between a Screen and a remote host. SunScreen also provides for the use of tunneling and VPNs.

SunScreen incorporates cryptography at the network (IP) layer to provide privacy and authentication over insecure public networks such as the Internet. For full descriptions of SKIP, IPsec, and the Sun Certification-Authority-issued keys and certificates, see the *SunScreen SKIP User's Guide, Release 1.5.1*, "Overview of IPv6" in *System Administration Guide, Volume 3*, and "Overview of IPsec" in *System Administration Guide, Volume 3*.

You can use the `skiptool` GUI, IKE GUI, or the command line to set up an Administration Station to encrypt administration commands that travel from a remote Administration Station over a potentially insecure network to a Screen.

How SunScreen Uses Encryption

SunScreen uses a combination of public-key and shared-key cryptography to encrypt and decrypt packets. Any traffic that passes between any two systems or other SKIP devices can be encrypted. All administrative traffic between a Screen and a remote Administration Station is encrypted.

Packet Examination

When the rules and policies determine that a specific packet should be encrypted, the encrypted packet is first checked to see if it is too large to forward. Encrypted packets can be larger than the original packet for the following reasons:

- The original packet is encapsulated in a new IP packet for transmission over the Internet.
- A SKIP or IPsec header is added so that the receiver can decrypt the packet.
- The encryption process requires some padding of the original data.

If the new packet is too large to send on and the original packet carries the "Don't Fragment" bit, then a message is sent back to the sender with a request for a smaller packet: "ICMP Fragmentation needed, but Don't-Fragment bit set." If the new packet

is too large and fragmentation is allowed, the original packet is first fragmented and then encrypted. The other end of the encryption tunnel can then decrypt each fragment independently.

The encryption routine builds a new IP packet to carry the original data. It uses the original source and destination addresses, or if tunneling is specified, the tunnel source and destination addresses.

After a new packet is created, the original packet is encrypted using the encryption mechanism specified (such as DES, RC2, or RC4) and a randomly generated traffic key. The traffic key is then encrypted using the encryption mechanism specified (DES or RC2) and a key-encrypting key from the SKIP key manager. The new IP header, SKIP header, and encrypted data are concatenated to form the new IP packet that is sent on to the destination addresses.

Note – Because of a limitation in SunScreen SKIP 1.5.1 for Solaris, the RC2 encryption algorithm is not available when running Solaris 7 and Solaris 8 in 64-bit mode.

When an encrypted packet is received, it is passed to the decryptor. By examining the SKIP header, the decryptor determines the correct decryption mechanisms for both the encrypted traffic key (DES or RC2) and the encrypted data (DES, RC2, or RC4). It retrieves the traffic-encrypting key from the key manager and decrypts the traffic key that in turn decrypts the original IP packet.

Finally, the decrypted packet is sent through the rules or policies to determine the action for the packet (for example, whether the decrypted packet should be passed or dropped).

Note – See “Using IKE With SunScreen” on page 93 for the IKE case.

Tunneling

SunScreen can use encryption in a feature called *tunneling* that is used to hide actual source and destination addresses. With tunneling you can substitute other addresses for the addresses on the packet header. When the Screen tunnels a packet, it replaces the packet’s source address with the *tunnel address* of the From Encryptor and replaces the packet’s destination address with the (optional) tunnel address of the To Encryptor. When the Screen decrypts a packet, the original addresses are restored.

Organizations typically have offices in more than one location. The tunneling feature enables the different offices to use public networks as a secure private network without needing dedicated lines and with no changes to user applications.

When a tunnel, or *virtual private network* (VPN), is set up between two locations, all data packets traveling from one location to the other are encrypted and encapsulated inside other packets before they are sent over the public internetwork. Encrypting the packets guarantees that their contents remain private; anyone capturing packets to snoop on network traffic between the two locations will be unable to read them. When the packets arrive at the remote location, they are extracted, decrypted, and forwarded to their intended destination.

In addition to protecting the privacy of network traffic, tunneling also lets a site conceal the details of its network topology from intruders or eavesdroppers. Because the original packets are encrypted, the source and destination addresses in their IP headers cannot be read. When the encrypted packets are encapsulated in other packets, the new IP headers identify the addresses of the Screens that protect the locations, not the hosts that originated the packets. Consequently, the network topology behind the Screens is never exposed.

The figure below illustrates how tunneling affects the outside IP header of a packet traveling from Host A to Host B. Note that the encrypted packet containing the original data Host A sent to Host B is the same whether or not tunneling is used. Note also that the addresses in the figure are not meant to be valid IP addresses; they have been simplified for illustrative purposes.

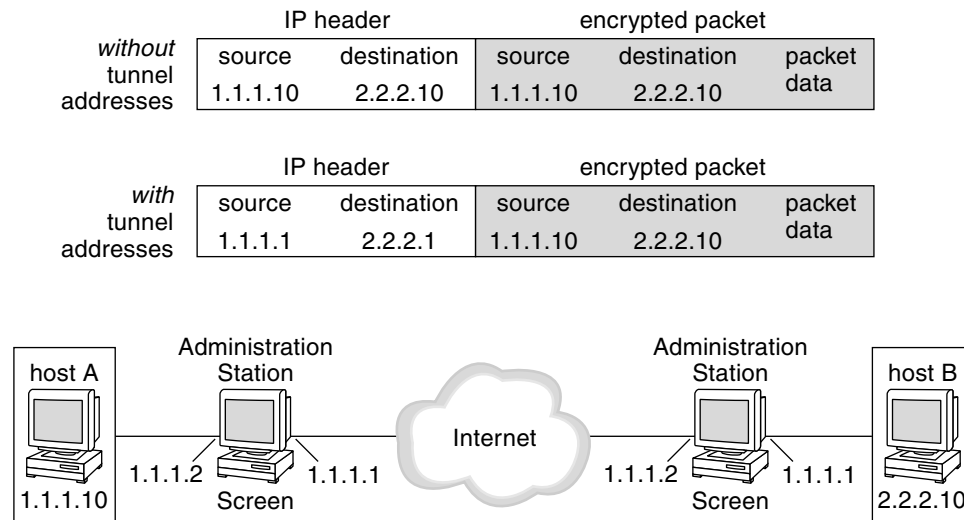


FIGURE 6-1 Effect of Tunneling on Contents of IP Headers

If you do not use a VPN (a tunnel) to transfer data between the two locations, the source and destination addresses on the outer IP header are the same as the source and destination addresses on the inner (encrypted) IP header: 1.1.1.10 and 2.2.2.10, respectively. Someone intercepting this packet cannot read its contents, but can

determine that hosts with IP addresses 1.1.1.10 and 2.2.2.10 reside behind the Screens protecting the two locations.

If you use a VPN to transfer data between the two locations, the Screen protecting Host A substitutes the IP address of its external interface (1.1.1.1) for the IP address of host A (1.1.1.10) in the Source Address field of the external IP header. Similarly, it substitutes the external IP address of the Screen at the other end of the tunnel (2.2.2.1) for the IP address of Host B in the Destination field. The local Screen then sends the packet through the insecure public network to the remote Screen.

When the remote Screen receives the packet, it strips off the encapsulation and decrypts the original packet from Host A. The remote Screen then reads the destination address from the original IP header of the decrypted packet and forwards the packet to Host B.

Because the IP addresses behind the Screen are never exposed to hosts on the external Internet, the two locations do not need to assign externally valid IP addresses to hosts behind the Screens. As long as hosts behind the Screens do not need to communicate with hosts on the Internet, the two locations can use a shared IP address space, as shown in Figure 6–2.

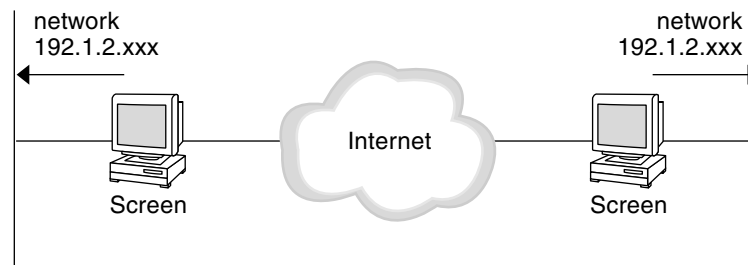


FIGURE 6–2 Geographically Dispersed Network

Using IKE With SunScreen

This section describes the IKE syntax and options as well as providing command line examples of policy rules that use IKE. You can find detailed administration GUI and command line instructions for using IKE in the *SunScreen 3.2 Administration Guide*. Additionally, the *SunScreen 3.2 Configuration Examples* manual has examples of using IKE for encryption.

IKE usage within SunScreen had three components: the IKE negotiation, the authentication header (AH), and the encryption header (ESP). Either the AH or ESP

option can be omitted, but authentication, which is available from both AH and ESP, must be included. Possible combinations are:

- IPSEC AH(authalg1) IKE(*IKE options*)
- IPSEC ESP(encralg1) IKE(*IKE options*)
- IPSEC ESP(encralg1, authalg2) IKE(*IKE options*)
- IPSEC AH(authalg1) ESP(encralg1) IKE(*IKE options*)
- IPSEC AH(authalg1) ESP(encralg1, authalg2) IKE(*IKE options*)

Note – Unlike SKIP syntax, the IPsec and IKE parameter lists use parentheses to contain them.

Possible values for authalgN are:

- MD5
- SHA1

Possible values for encralgN are:

- 3DES-CBC
- AES
- BLOWFISH
- DES-CBC
- NULL

The NULL algorithm is generally used for testing because it exercises nearly all the normal code paths but does not obscure the data. That means what is inside can be easily seen. In general, unless a non-NULL encryption algorithm (like DES-CBC or 3DES-CBC) is applied, the data will not be obscured.

The AH and ESP options control the cryptographic means that are used to protect the DATA portions of network traffic. They are functional equivalents of the DATA and MAC algorithms used in SKIP.

The IKE option performs the functional equivalent of the rest of the options in SKIP, including the KEY algorithm and the naming of the certified cryptographic data to be used for configuring and securing the traffic.

IKE Options

IKE options have the form:

`IKE(encralg2, authalg3, oakleygroup, PRE-SHARED, presharedkey)`

or

IKE(encralg2, authalg3, oakleygroup, authmethod, srcidentity, dstidentity)

Note – The above syntax is that used in policy rules. In the contexts of AccessRemote, Screen, and VPNgateway objects, the syntax does not allow the PRE-SHARED formulation, the srcidentity is the local Screen's identity, and there is no dstidentity value.

For IKE, the parameters given determine the mechanisms to be used to validate signed security items, and the algorithms and parameters to be used to negotiate keys and other interactions which precede the actual transmission of data. (In IKE parlance, this is called the "phase 1" negotiation; "phase 2" is the use of the negotiated key to secure the client data.)

The lists for encralg2 and authalg3 are the same as for AH and ESP (see the third component listed for IKE usage within SunScreen).

The oakleygroup parameter represents a Diffie-Hellman Group. That parameter controls the type of cryptographic mathematics to be used in key generation. These are given as single-digit numbers. SunScreen supports:

- 1 - 768-bit Diffie-Hellman modulus
- 2 - 1024-bit Diffie-Hellman modulus
- 5 - 1680-bit Diffie-Hellman modulus

The authmethod determines how the certified key items (for example, certificates) are to be validated. The current values are:

- DSS-SIGNATURES
- RSA-SIGNATURES
- RSA-ENCRYPTION
- PRE-SHARED

IKE Certificates

SINGLE IKE certificates contain a matching pattern, or even a portion of a matching pattern, that is evaluated as needed by the IKE software. IKE certificates have a variety of naming methodologies, among them DNS names of hosts, mailbox names of users (which contain DNS names), IP addresses (both V4 and V6), and X.500 composite names.

IKE certificate groups are also dissimilar from SKIP groups. In IKE, a certificate group is defined in a manner similar to that of SunScreen address objects (with some restrictions). The IKE certificate group is really just a mechanism for expressing composite names (complex patterns).

The restrictions on IKE certificate groups relate to the context in which their exclusion lists can be used. Only a top-most IKE certificate group can use exclusions; all other groups can only contain inclusions. This restriction helps avoid various bizarre naming situations that might otherwise arise.

Pre-Shared Option

The PRE-SHARED option is a degenerate key certification mechanism. This option indicates that a manual key has been defined out-of-band between the peer systems, and that key requires no further validation. It differs from purely manual keying in that only the IKE negotiation uses the manual key; IKE still negotiates and changes the keys used for data protection.

The `srcidentity` and `dstidentity` certificates refer to Certificate objects. Certificate objects so used can be either SINGLE IKE or groups of other IKE Certificate objects.

Note – You cannot intermix IKE and SKIP certificate objects within a certificate group.

Certificate Options

`srcidentity` refers to the certificate(s) to be associated with and representing the source addresses with respect to the data protected by the policy rule in question.

`dstidentity` refers to the certificate(s) to be associated with and representing the destination addresses with respect to the data protected by the policy rule in question.

`srcidentity` and `dstidentity` must both be verifiable using the `authmethod` given.

IKE Policy Rules

Besides the IKE options given above, IKE policy rules can also specify a `SOURCE_SCREEN` and a `DESTINATION_SCREEN` clause. These clauses each take the name of a Screen object, and anchor the usage of a rule to particular Screens upon which to perform the specified IKE processing.

Like SKIP, IKE policy rules can also use the `SOURCE_TUNNEL` and `DESTINATION_TUNNEL` options; as in SKIP, these specify a (possibly fictitious) IP address (object) to be used as the tunnel identity at one or both ends of an encrypted tunnel.

In addition, IKE policy rules can use TRANSPORT mode, which does not tunnel the data (wrap a new IP header outside an inner one), but rather secures the data portion of an IP datagram, using and leaving exposed the original source and destination IP addresses.

IKE Policy Rule Syntax

Command line syntax for various IKE policy rules is shown below. Note that the backslash (\) at the end of a line indicates that the line continues on the next line. Do not include any Returns, Enters, or backslashes when typing rules.

- Tunnel mode, pre-shared key usage:

```
[SCREEN scrn] svc srcaddr dstaddr \
IPSEC { AH(authalg1) | ESP(encralg1[, authalg2]) } \
IKE(encralg2, authalg3, oakleygroup, PRE-SHARED, pskey) \
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \
[SOURCE_TUNNEL srctunaddr] [DESTINATION_TUNNEL dsttunaddr] \
ALLOW
```

- Tunnel mode, certificate usage:

```
[SCREEN scrn] svc srcaddr dstaddr \
IPSEC { AH(authalg1) | ESP(encralg1[, authalg2]) } \
IKE(encralg2, authalg3, oakleygroup, authmethod, \
srccert, dstcert) \
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \
[SOURCE_TUNNEL srctunaddr] [DESTINATION_TUNNEL dsttunaddr] \
ALLOW
```

- Tunnel mode, manual key usage:

```
[SCREEN scrn] svc srcaddr dstaddr \
IPSEC { AH(spi1, authalg, key1) \
| ESP(spi2, encralg2, key2 [, spi3, authalg3, key3]) } \
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \
[SOURCE_TUNNEL srctunaddr] [DESTINATION_TUNNEL dsttunaddr] \
ALLOW
```

- Transport mode, pre-shared key usage:

```
[SCREEN scrn] svc srcaddr dstaddr \
IPSEC { AH(authalg1) | ESP(encralg1[, authalg2]) } \
IKE(encralg2, authalg3, oakleygroup, PRE-SHARED, pskey) \
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \
TRANSPORT ALLOW
```

- Transport mode, certificate usage:

```
[SCREEN scrn] svc srcaddr dstaddr \
IPSEC { AH(authalg1) | ESP(encralg1[, authalg2]) } \
IKE(encralg2, authalg3, oakleygroup, authmethod, \
srccert, dstcert) \
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \
```

```
TRANSPORT ALLOW
```

- Transport mode, manual key usage:

```
[SCREEN scrn] svc srcaddr dstaddr \  
IPSEC { AH(spi1, authalg, key1) \  
        | ESP(spi2, encralg2, key2 [, spi3, authalg3, key3]) } \  
[SOURCE_SCREEN srcscrn] [DESTINATION_SCREEN dstscrn] \  
TRANSPORT ALLOW
```

Defining a VPN

A VPN is a group of Screens that transfer encrypted data among themselves. A VPN simulates a private network using a public network, with IP-level encryption providing privacy.

Note – A VPN object in SunScreen—called a “VPN” in this document—is not a virtual private network as generally defined in the firewall industry. It is a mechanism for implementing SunScreen’s version of a VPN.

After a VPN has been defined, you can refer to it when adding rules to your security policy. This means you can define your security policy with fewer rules. The system automatically generates the multiple rules that the VPN defines.

In defining a VPN:

- Choose a name. This name is used in the Name field in the VPN gateway entries. It is also used in any policy rules that refer to this VPN.
- Define a VPN gateway object for each Screen in a VPN. You define VPN gateway objects in the administration GUI through the VPN tab in the Policy Rules page.

When defining a VPN gateway object, which is a list of Screens, specify the following information:

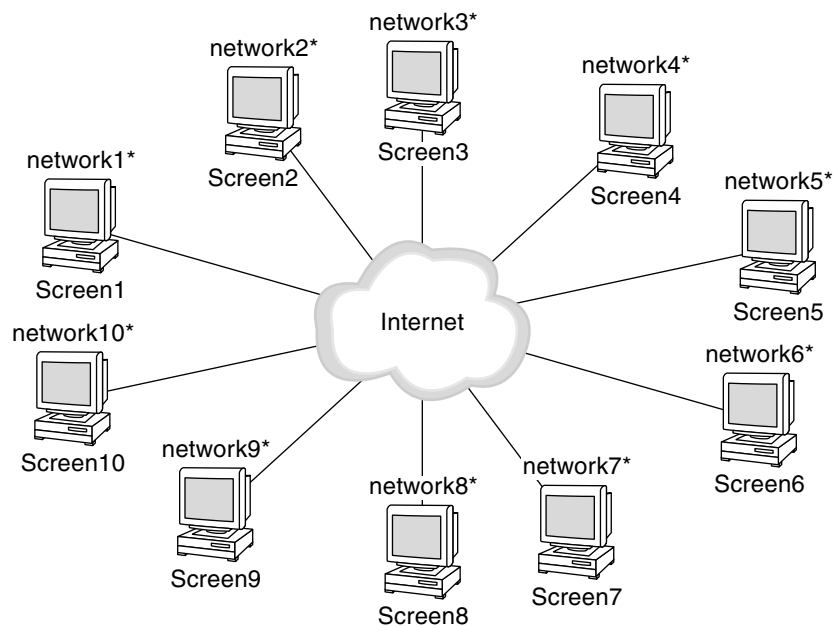
- Rule Index (Optional) – Assigns a number to the VPN gateway entry. This affects the position within the VPN gateway list. By default, the GUI will place new entries at the end of the current list. Because SunScreen uses ordered rules, be sure to place the rule in the order in which you want it to take effect. See “Stateful Packet Filtering” on page 46 for a description of ordered rules within SunScreen.
- Name – The name of the VPN of which this VPN gateway is a member. Use the name you chose for the VPN.
- Address – The addresses protected by this Screen. Generally, this address will be the same as one of the interface addresses for this Screen.

Note – You can use any address in the VPN rules. Only addresses that interact with a VPN gateway and the address specified in the rule will apply. The simplest rule uses * for the source and destination address. This rule allows encrypted use of the specified service for all addresses in the VPN.

- **Certificate** – The certificate used for this Screen when encrypting packets to other Screens in the VPN. For a particular VPN, all certificates must refer to keys of the same strength (for example, 512-, 1024-, 2048-, or 4096-bit Diffie-Hellman keys).
- **Key Algorithm** – The key algorithm that is used when encrypting packets to other Screens in the VPN. This field must be identical in all VPN gateway entries with the same VPN name.
- **Data Algorithm** – The data algorithm that is used when encrypting packets to other Screens in the VPN. This field must be identical in all VPN gateway entries with the same VPN name.
- **MAC Algorithm** – The MAC algorithm that is used when encrypting packets to other Screens in the VPN. This field must be identical in all VPN gateway entries with the same VPN name.
- **Tunnel Address** – The Screen's tunnel address that is used when encrypting packets to other Screens in the VPN.
- **Description** – Optionally, provide a short description of this VPN gateway entry.

Note – See `ssadm-rule (1m)` for information about VPNs using IPsec/IKE.

The site shown in Figure 6-3 has ten Screens. One of the systems protected by each Screen is a mail server. Assume that your security policy allows the exchange of encrypted mail between all these mail servers and you want to define rules to allow SMTP between all of the mail servers.



* Each network contains a mail server, named MailServer1 through MailServer10 respectively.

FIGURE 6-3 Sample Ten-Network Site

Without a VPN, you must define nine rules on each Screen for each mail server to send mail encrypted to the other nine mail servers. Because you have ten mail servers, you must define a total of 90 rules. If, instead you defined a VPN, you only need a single rule: one that allows the mail servers to send mail to the other mail servers using the VPN. Because you have ten Screens in the VPN, you must define ten VPN gateway entries.

Looking at this example in detail, the figure below shows the configuration. In this example, the name for the VPN is "ourVPN." The Screens are labeled Screen1 through Screen10. The mail servers behind them are labeled mail1 through mail10 and are part of network1 through network10.

Once you have defined the VPN objects, as shown in the figure below, you can use the VPN in any rule. Select VPN as the action for the data between the sender and the SMTP server, and specify the name of the VPN.

Policy Rules								
Policy Name: pol4 version 5								
<div> Packet Filtering Administrative Access NAT VPN </div>								
Rule Index	Name	Address	Certificate	Key Algorithm	Data Algorithm	MAC Algorithm	Tunnel Address	Description
1	ourVPN	Network1	Certificate1	DES-CBC	DES-CBC	MD5	Tunnel1	VPNmarketing-
2	ourVPN	Network2	Certificate2	DES-CBC	DES-CBC	MD5	Tunnel2	VPNmarketing-
3	ourVPN	Network3	Certificate3	DES-CBC	DES-CBC	MD5	Tunnel3	VPNmarketing-
4	ourVPN	Network4	Certificate4	DES-CBC	DES-CBC	MD5	Tunnel4	VPNmarketing-
5	ourVPN	Network5	Certificate5	DES-CBC	DES-CBC	MD5	Tunnel5	VPNmarketing-
6	ourVPN	Network6	Certificate6	DES-CBC	DES-CBC	MD5	Tunnel6	VPNmarketing-
7	ourVPN	Network7	Certificate7	DES-CBC	DES-CBC	MD5	Tunnel7	VPNmarketing-
8	ourVPN	Network8	Certificate8	DES-CBC	DES-CBC	MD5	Tunnel8	VPNmarketing-
9	ourVPN	Network9	Certificate9	DES-CBC	DES-CBC	MD5	Tunnel9	VPNmarketing-
10	ourVPN	Network10	Certificate10	DES-CBC	DES-CBC	MD5	Tunnel10	VPNmarketing-
<div> Add New... Edit... Move... Rename... Delete ... Help... </div>								

FIGURE 6-4 VPN Tab with VPN Entries

Adding a VPN Rule

Assuming an address group named MailServers containing all the mail servers exchanging encrypted mail, define the rule in the Rule Definition dialog box shown in the figure below.

Rule Index	1
Screen	*
Service	smtp
Source Address	MailServer
Destination Address	MailServer
Action	VPN
Time	*
Description	VPNmarketing

Show Action Details

OK Cancel

Unsigned Java Applet Window

FIGURE 6-5 Completed Rule Definition Dialog Box for the VPN Rule

The VPN rule appears on the Packet Filtering tab of the Policy Rules page. The more restrictive a rule is, the earlier it should be ordered in the list of rules because the rules take effect in order. The more restrictive VPN rule comes before the more general rule and so will take effect earlier.

[illegible]**FIGURE 6-6 VPN Rule**

There is no limit to the number of VPNs to which a Screen can belong. For example, you can define two VPNs—one for encryption at 1024 bits, and one for encryption at 4096 bits. A single Screen can belong to both of those VPNs: one entry specifying the 1024-bit certificate, and the other specifying the 4096-bit certificate.

VPN Limitations

Currently, the VPN object has the following limitation:

- The key, data, and MAC algorithms must be the same for all gateways within a VPN.

Network Address Translation

For a system to communicate with other systems on the Internet, it must have an IP address—a unique 32-bit number that identifies the system. The rapid growth of the Internet has brought about a shortage of IP addresses. Network address translation (NAT) provides a solution for this shortage. This chapter contains information on:

- “Network Address Translation” on page 105
- “NAT Rules” on page 107
- “Static NAT” on page 107
- “Dynamic NAT” on page 109
- “Dynamic NAT Collisions” on page 110
- “Choosing NAT Addresses” on page 112
- “NAT Examples” on page 112
- “Applying NAT” on page 116

Network Address Translation

Network address translation enables a Screen to translate an internal network address to a different public network address. As it passes packets between an internal host and a public network, the addresses in the packet are replaced with new addresses transparently, checksums and sequence numbers are corrected in both the IP header and the TCP or UDP header, and the state of the address translation is monitored. You specify when a packet using ordered NAT is applied based on source and destination addresses.

SunScreen NAT gives you fine-grain control by adding ordered NAT translations, allowing table entries to intersect, and enabling you to specify when to have NAT translate the source or destination addresses.

Note – Be sure to configure your NAT rules so they do not perform address translation while an internal host is attempting to communicate directly with the Screen

Services such as FTP also carry IP address information. These packets must also be changed, ensuring that the checksums and sequence numbers are correct. All of this is done inside the Screen's kernel to ensure high-speed processing and transparency to the end user and applications. NAT is stateful, which increases the efficiency of lookups in the address translation table by using address hashings and checksum adjustments that use differential checksum calculations.

NAT is typically used when:

- Renumbering all the hosts in your network is not feasible.
- The current private network uses a set of unregistered (private) IP addresses owing to a lack of available public addresses, or to simplify renumbering hosts should you change ISPs.
- You have a very large network to connect, but your ISP allotted you a limited range of IP addresses.
- You want to hide the addresses on the current private network from the outside world.

Note – Using NAT to hide addresses differs from tunneling in that NAT does not rely on the use of encryption and decryption and, consequently, does not require an encryption-decryption device or software at each end of the connection over the public network.

NAT lets you use unregistered Internet addresses to number your internal networks and hosts and still maintain full connectivity to the Internet. The Internet Assigned Numbers Authority (IANA) has reserved the following three blocks of the IP address space for private internets:

- 10.0.0.0–10.255.255.255 (Class A address range, which supports about 16 million addresses)
- 172.16.0.0–172.31.255.255 (Class B address range, which supports about 65,000 address)
- 192.168.0.0–192.168.255.255 (Class C address range, which supports 254 addresses)

With this approach, you can use a registered (public) Class C address space, which offers about 254 externally routable addresses, to provide connectivity for an unregistered Class B network, which supports approximately 65,000 hosts or 255 networks of 254 hosts (internally).

NAT Rules

There are two types of NAT rules: *static* and *dynamic*. Static NAT rules provide a one-to-one translation of addresses. Dynamic NAT rules provide an N-to-M, typically many to one, translation of addresses.

NAT rules are specified by their type (STATIC or DYNAMIC), their source and translated source address, and their destination and translated destination address. Addresses in NAT rules use the same set of address objects used in other rules.

NAT rules are ordered. The first NAT rule that matches a packet takes effect, and no other NAT rules apply. Therefore, place specific NAT rules first, and broader NAT rules later.

Valid translations are:

- Source and Translated Source are the same, and Destination and Translated Destination are the same (no translation occurs).
- Source and Translated Source are the same, and Destination and Translated Destination are different.
- Source and Translated Source are different, and Destination and Translated Destination are the same.

You cannot translate both the source and destination addresses in any single packet with either a single translation or in a combination of translations. A nontranslating NAT rule may be placed ahead of more general NAT rules, to override part of the later, more general NAT rule.

Static NAT

Registered addresses are necessary for *advertised* kinds of resources, such as publicly accessible servers on your network, because these machines must be at well-known, fixed addresses. Static NAT is frequently used to provide public access to HTTP or FTP servers that use private addresses. These servers must use static NAT reverse rules so that other hosts can use the same registered addresses to reach them. You must generate the reverse rules.

One-to-One Translations

Use static NAT rules to make one-to-one translations between either a single pair or multiple pairs of addresses. Most commonly, static NAT rules are used to translate an advertised address for a public server to a different address.

A static NAT rule translates either the source or destination addresses in a packet. In most cases, this means that you will need to define two NAT rules to:

- Translate the source address when the packet is flowing in one direction.
- Translate the destination address when packets are flowing in the other direction.

As an example of static NAT rules in one-to-one translation, assume that your public web server has an address of 10.0.0.1 (defined by the address object "private_www") and you want to allow access to this web server through the public address 199.190.177.1 (defined by the address object "public_www"). Assume also that the address Internet represents Internet addresses. To do this requires two static NAT rules, as shown in the table below

- The first rule specifies that the destination address `public_www` (199.190.177.1) is the translated destination address `private_www` (10.0.0.1). This NAT rule handles packets flowing *to* the web server.
- The second rule specifies that the source address `private_www` (10.0.0.1) is the translated source address `public_www` (199.190.177.1). This NAT rule handles packets flowing *from* the web server.

TABLE 7-1 Static NAT Rules

Type of NAT Rule	Source	Destination	Translated Source	Translated Destination	Comment
STATIC	"Internet"	"public_www"	"Internet"	"private_www"	Packets to server
STATIC	"private_www"	"Internet"	"public_www"	"Internet"	Packets from server

Address Range to Another Address Range

You also can use Static translations to translate a range of unregistered addresses to a range of registered addresses. Each range of addresses must contain the same number of addresses.

- Example of Static NAT Rule with a Range of Addresses
You can translate the address range containing 199.190.177.1 through 199.190.177.100 to an address range containing 10.0.0.1 through 10.0.0.100 because both ranges contain 100 addresses. In this example, 199.190.177.1 translates to 10.0.0.1, 199.190.177.2 translates to 10.0.0.2, ending with 199.190.177.100 translating

to 10.0.0.100.

Dynamic NAT

Use dynamic NAT to translate a set of unregistered IP addresses to a smaller set of registered addresses. Dynamic NAT enables you to connect to a large number of hosts to the public Internet using a limited number of registered addresses.

Unlike static NAT, which sets up a one-to-one translation between internal unregistered addresses and external registered addresses, dynamic NAT creates a many-to-one translation where several internal addresses use the same public address. Dynamic NAT avoids IP address conflicts by maintaining a state table that records five values (source address, source port, destination address, destination port, and protocol) for each TCP or UDP connection. A Screen can multiplex thousands of translations over a single registered address.

Dynamic NAT is unidirectional, meaning that communication can be initiated only internally from the unregistered private network. Dynamic NAT only works when a user originates a connection from inside the firewall; packets from outside that are not in the address lookup table of an established connection cannot identify a host on the private network and are discarded. Dynamic NAT only works for connections initiated from the Source address systems. These generally represent machines with unregistered addresses that you want to translate to registered address.

For example, assume you have workstations with unregistered addresses defined by the address group `my_private` that you want to allow access to the Internet using a set of public addresses defined by the address group `my_public`. The address `Internet` represents the addresses of the internet.

To do this, you define a dynamic NAT rule, as shown in the table below, that specifies that the Source address `my_private` becomes the translated source address group `my_public`. Destination and translated destination are the address `Internet`, which limits the scope of the translation of packets going to or from the Internet.

TABLE 7-2 A Dynamic NAT Rule

Type	Source	Destination	Translated Source	Translated Destination	Comment
DYNAMIC	my_private	Internet	my_public	Internet	

Dynamic NAT Collisions

Because dynamic NAT translates a large set of addresses into a smaller set of addresses, the addresses could be translated to the same address and port numbers, in which case the translations are said to collide.

Note – The chance of such a collision is very small.

Address collisions occur if the Screen cannot translate the address uniquely. An address collision causes the connection to cease. Address collisions occur if *all* the following conditions are met:

- Two systems using NAT connect to the same destination service (the same remote address and the same remote port number using the same protocol).

For example, this condition is met if two systems using NAT establish a web connection to `www.sun.com`.

- The two systems choose the same local port number to make the connection.

Because most systems select from a set of at least 32 000 different local port numbers, the chance of this happening is usually small.

- The NAT code chooses to translate the two connections using the same translated address.

Expressed as a probability, the chance of this happening for two systems is equal to $1/M$ where M is the number of addresses in the Translated Source field. For example, if the Translated Source field contains an address object that represents 10 addresses, the probability of NAT choosing the same translated source address for two systems would be $1/10$ or 10 percent.

The probability of a collision is equal to the probability of two systems connecting to the same remote service *times* the probability of two systems choosing the same local port *divided by* the number of addresses in Translated Destination.

Translation collisions cause service to be denied to a network user. Translation collisions occur when network software cannot complete the address translation process because two or more packets are not uniquely identified. Each packet must have a destination IP address, a destination port, source IP address, a source port, and protocol if it is to be delivered. These elements are processed as a *5-tuple* of information of the form (desaddr, dest port, srcaddr, src port, proto), which is part of the packet header.

A 5-tuple is unique as long as at least one of the five pieces of data that it contains differs from the others with which it is being compared. Since each piece of data has a large number of possible values, the number of possible permutations for the 5-tuple is enormous. Therefore, for a translation collision to occur, multiple internal machines using the same registered IP address must try to gain access to the same registered address at the same destination port number and from the same source port number, all at the same time.

Suppose a user at the unregistered address U5, shown in Table 7–3, attempts to go to a web page at the registered destination address 192.4.15.37 at destination port 80 from source port 34080 through the registered address R5. Another user at U6 can do the same to the same address and destination port through source port 34070, or go to a different web page through source port 34080.

The table below shows the translation of unregistered addresses, Un , to registered addresses, Rn .

TABLE 7–3 Two Dynamic Addresses

Registered IP Address	Destination IP Address	Destination Port	Source Port	Protocol
R4	192.4.15.37	80	34080 (on U5)	tcp
R4	192.4.15.37	80	34070 (on U6)	tcp
R4	192.4.15.44	80	34080 (on U7)	tcp

If a user at unregistered address U7 attempts to go to a web page at the registered destination IP address 192.4.15.44 at destination port 80 from source port 34080 using registered address R4, a translation collision will occur. The user at U7 would have to use another source port to have a unique 5-tuple and avoid a translation collision, which would happen automatically during a subsequent attempt to connect.

Situations such as power failures typically result in translation collisions. When power is restored, all hosts on a network come up at the same time and try to reestablish network connections. Each host's operating system resets its source port counter to a low number. The counters on each machine may take time to cycle up to higher and more randomized port numbers (which are more likely to produce unique 5-tuples). In the interim, translation collisions may cause network service to be denied temporarily. Internal hosts must continue trying to establish network connections until the NAT rules resolve the translation collisions.

Note – Ports 0 through 1024 are reserved for well-known port assignments and are controlled by the IANA. To avoid conflicts, the Solaris operating environment uses ports that range approximately from 32768 through 65535. Different implementations of TCP/IP in various operating environments have different rules and limits for their optional (ephemeral) port choices.

Choosing NAT Addresses

In all NAT situations, one of the addresses in a NAT translation is virtual. It does not really exist and must be simulated by the Screen and other systems in the network.

- If Source and Translated Source addresses are different, then the Translated Source address is virtual.
- If Destination and Translated Destination addresses are different, the Destination address is virtual.

Because virtual addresses do not physically exist, how these address are selected and who simulates them is restricted. Simulating a virtual address means providing the functions of ARPing and routing.

In routing mode, the Screen must respond to ARP requests for the public addresses (R2 through R8) because it will be translating public addresses to private addresses. Add an `arp` entry (using the Solaris command `arp -s IP_address ether_address pub`) for them. Either add this entry each time that you reboot the Screen or add it to a startup script that runs at boot time. If you are administering the Screen in routing mode remotely, either go to the Screen to add this entry, or have a rule in your policy that allows logging in (`rlogin`) to the Screen remotely.

In stealth mode, the Screen automatically responds to the ARP requests from a public address so the ARP entry is not necessary.

NAT Examples

The following NAT examples show how to set up NAT when using only one registered IP address, and with two scenarios that illustrate how a demilitarized zone could use registered addresses or unregistered addresses with NAT.

Example One

If you only have one registered IP address (A) and you want to have all inbound traffic go to A, go to your Screen and have all other hosts use that address (A) for unidirectional, outbound traffic. Then set up NAT as shown in the table below.

TABLE 7-4 Example of a One-Address NAT Table Entry

Index	Screen	TYPE	Source	Destination	Translated Source	Translated Destination	Comment
1		STATIC	*	A	*	A	
2		DYNAMIC	Inside	Internet	A	Internet	

Internet is all addresses on inbound interface A; and Inside is all internal hosts on all other interfaces. With only these NAT rules, all hosts in the Inside communicate with their private, unregistered addresses when communicating with the Screen or among themselves.

Write your filtering rules in the context of the internal addresses.

Example Two

Registered addresses are necessary for *advertised* kinds of resources, such as publicly accessible servers on your network; consequently, these machines must be at well-known, fixed addresses. Because a host must have a registered address before it can communicate over public networks, machines that host public resources either must have stable registered addresses, or their unregistered (internal) addresses must translate to stable registered addresses. The following scenarios illustrate how a demilitarized zone (DMZ), an internal network with limited public access, could use registered addresses or unregistered addresses with network address translation.

Scenario 1: DMZ Uses Registered Addresses

In the figure below, the Screen, in routing-mode, uses Q1 as its own IP address on the external network interface. It has a DMZ network with registered addresses R1 through R8 on a second interface. The Screen (Q1) and the servers in the DMZ (the FTP server, R2, and the WWW server, R3), have routable registered addresses on the public network that allow them to communicate with any other machine with a registered address. The Screen uses the remaining registered addresses (R4 through R8) for NAT.

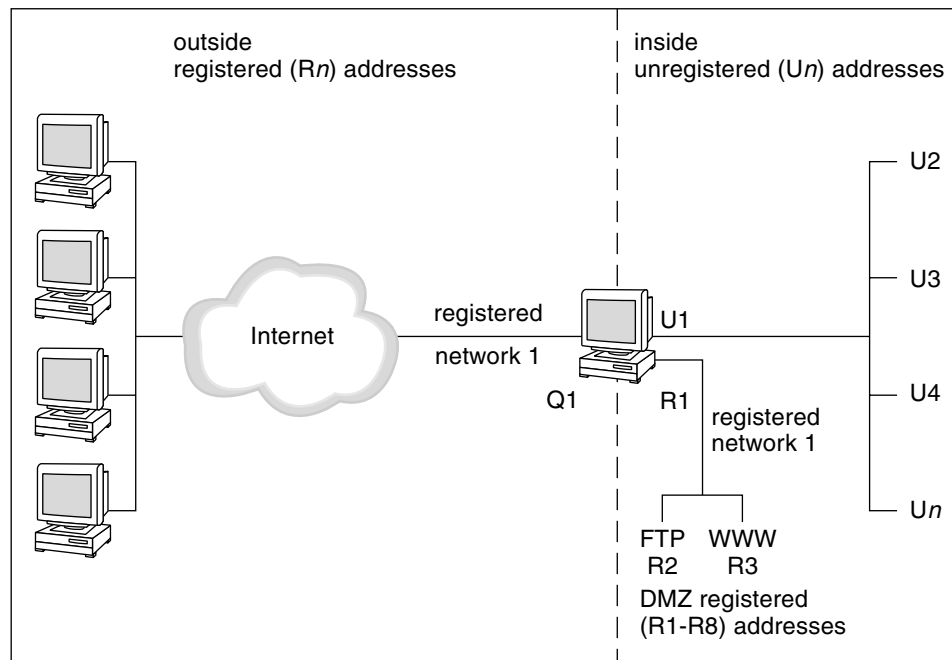


FIGURE 7-1 Scenario 1: Static and Dynamic NAT

The Screen uses dynamic NAT to translate the addresses in its unregistered address range (U_2 – U_n) to the remaining addresses in its registered address range (R_4 – R_8). When an internal host with an unregistered address tries to connect to an external host with a registered address, the Screen assigns the internal host a registered address to use for the duration of the network communication session.

Scenario 2: DMZ Uses NAT Addresses

The figure below illustrates an organization that has a network consisting of a large number of unregistered addresses (U_n) and a set of eight registered addresses (R_1 – R_8). Hosts on the inside network must be able to communicate through the Screen with external hosts.

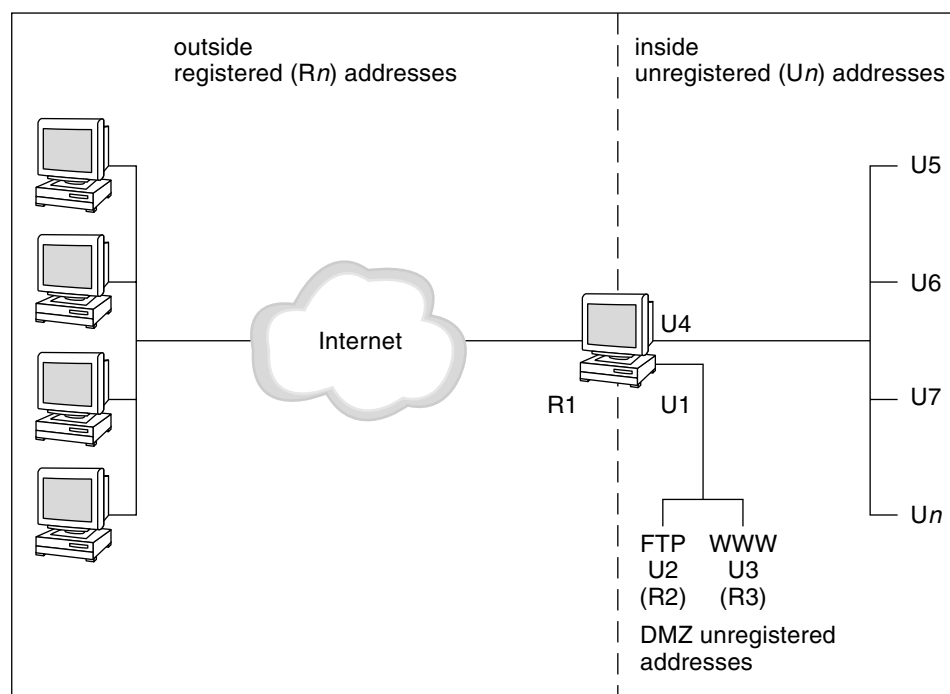


FIGURE 7-2 Scenario 2: Static and Dynamic NAT

In the figure above, the Screen is connected to the public network R1-R8. R1 is its IP address on the public network interface. It uses static NAT to translate the unregistered DMZ addresses of the FTP server (U2) and the WWW server (U3) to the registered (public) addresses R2 and R3. The private addresses U4 through U_n will be translated dynamically to the registered addresses R4 through R8. Because the IP addresses of the servers and the internal network are translated to routable registered addresses, they can communicate with any other registered address.

The Screen uses dynamic NAT to translate the addresses in its unregistered address range (U4– U_n) to the remaining addresses in its registered address range (R4–R8). When an internal host with an unregistered address tries to connect to an external host with a registered address, the Screen assigns the internal host a registered address to use for the duration of the network communication session.

This scenario has the advantage that if you change ISPs, you do not have to re-address all the hosts on your internal registered network.

Routing Interface Examples

For routing interfaces, you can select the registered address as the address of one of the Screen's interfaces. In this case, the Screen simulates the registered address. The limitation here is that you only have a single address. Also selecting the interface address as the registered address for a static NAT rule can limit your ability to connect to the Screen itself. Because you are not adding additional networks, no routing changes are required.

For routing interfaces, you can select the unused addresses on one of the networks to which the Screen is directly connected as virtual addresses. In this case, this approach is necessary so that the addresses can respond to ARP requests for these virtual addresses.

For routing interfaces, if you select the virtual addresses from a network not directly connected to the Screen, you must make sure that the correct routing information is propagated so that packets destined for these addresses pass through the Screen. If you define new networks (especially ones in which all the addresses on the network are virtual), you may need to add static routing entries on some routers to simulate these networks.

Stealth Interface Examples

For stealth interfaces, you can select the registered addresses from the list of unused addresses on the network that the Screen segments. In this case, Screen simulates the virtual addresses and responds to ARP requests for those addresses. Since you are not adding additional networks, no routing changes are required.

For example, consider a SunScreen with stealth interfaces that segments the network 199.190.177.0 (netmask 255.255.255.0). In this example, the addresses 199.190.177.100 through 199.190.177.254 are unused and can be used as virtual addresses in network address translations.

For stealth interfaces, you can select the registered addresses from a new virtual network you create. For this to work successfully, you must be able to assign multiple addresses on multiple networks on the routers you use.

Applying NAT

NAT translations for your site are automatically applied to all packets. When packets addressed to an internal host are received from an external host, the Screen translates

network address information in the packets before they are processed by the stateful packet-filtering engine. Similarly, packets travelling from an internal host to an external host are filtered before NAT takes place. This approach lets you use your internal addresses when you define filtering rules, simplifying policy management.

High Availability

High Availability (HA) provides redundancy if the hardware or software fails on the firewall system. HA works in routing, stealth, and mixed modes and it works with encryption. Failover can be automatic or manual. This chapter contains information on:

- “SunScreen HA Configurations” on page 121
- “HA Network Connections and Failovers” on page 125
- “Configuring HA” on page 126
- “Administering HA” on page 127

Why High Availability?

High Availability allows you to deploy groups of Screens together in situations where the connection between a protected inside network and an insecure outside network is critical.

Hardware Requirements

To configure SunScreen with HA, you need additional hardware for the secondary systems. For HA, you need at least one secondary system, but you can have as many as 31. The hardware for an HA configuration depends on the amount of traffic it is expected to filter and the overhead for processing the traffic. This overhead depends on the number of rules in a policy, encryption, and the like. All the hardware in the HA cluster, ideally, should be the same because they all must process the same volume of data. The network interfaces must be of the identical type (for example, `qfe`) and the have identical names (for example, `qfe0`).

SunScreen HA Definitions

One Screen in an HA configuration is defined as the *primary* Screen. The rest are defined as *secondary* Screens.

- **Primary HA Screen** – When you set up an HA cluster, you designate one Screen as its primary HA Screen. The primary HA Screen contains the editable configuration for the HA cluster. When you activate a policy on the primary Screen, its rules are copied from the primary HA Screen to all the secondary HA Screens in the HA cluster.

Solaris settings, such as network interfaces and routing configuration, are not copied from the primary Screen to the secondary Screens and must be identical on all the Screens in the HA cluster. The address of the HA interface on the primary Screen must be unique. The node name for the primary Screen must be unique.
- **Secondary HA Screen** – These Screens are the systems that do not have the editable configuration on them. They receive the configuration from the primary Screen. The interfaces must be the same physical type and have the same names as the primary Screen. For example, if the primary uses `le0` and `qe0` for filtering and `qe1` as the HA interface, the secondary must also use `le0` and `qe0` for filtering and `qe1` as the HA interface. The filtering interfaces must have the same IP address as the primary Screen. The address of each HA interface on each Screen must be unique. Similarly, the node name of each Screen must be unique.

At any time, one member of the HA cluster is the *active* Screen and the other Screens in the HA cluster are the *passive* Screens. When a configuration is activated, the primary HA Screen transfers the configuration, including certificates, local keys, addresses, policy rules, etc., to all Secondary HA Screens.

- **Active Screen** – The active Screen filters packets, translates network addresses, logs packets according to the action in a rule, and encrypts or decrypts packets. The active Screen can be a secondary Screen. Any Screen can become the active Screen.
- **Passive Screen** – The passive Screens receive the same packets and perform the same calculations as the active Screen, and mirror the configuration of the active Screen, but they do not forward traffic.

Under normal circumstances, the primary Screen is the active Screen. It receives, processes, and sends packets. All the secondary Screens are passive Screens. They receive and process, but do not send any packets. If the primary Screen fails for some reason, one of the secondary Screens becomes the active Screen. If the primary Screen subsequently becomes operational again while a secondary Screen is active, the primary Screen comes up as a passive Screen and is eligible to become the active Screen if the active Screen fails or is manually forced into passive mode. The primary Screen does not have to be the active Screen.

All the screens in your configuration must do name resolution through `/etc/hosts`.

The IP addresses of the HA heartbeat interfaces for each member of the HA cluster for dedicated network connections must be unique. Assign all HA Screens the same IP

addresses on their filtering interfaces. If a remote Administration Station connects to the IP address of one of the filtering interfaces, the active Screen will respond. The active Screen is not necessarily the primary Screen, which contains the policy. The Administration Station must use the IP address of the HA interface, if you want to be sure that it is connecting to the primary Screen.

SunScreen HA Configurations

You can configure in both routing and stealth mode. See “Routing and Stealth Mode Interfaces” on page 32 for more information.

Basic SunScreen HA in Routing Mode

The figure below shows a network protected by at least two identical Screens in an HA cluster. They are administered remotely.

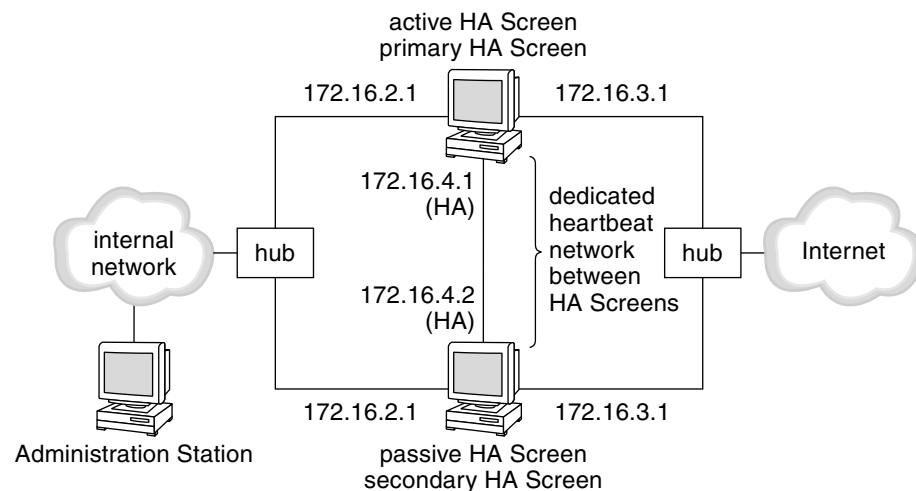


FIGURE 8-1 Network With HA Cluster of Screens

Each Screen in the HA cluster connects to the external and internal networks through Ethernet hubs, which pass the same packets to all members of the HA cluster at the same time.

The routing interfaces of all the systems in the HA cluster have the same interface names with the same IP addresses. When a Screen becomes a secondary Screen, the MAC address of its routing interfaces is changed so that it is the same as the MAC address of the corresponding interface on the primary Screen. Each HA Screen, therefore, receives the same traffic, ensuring that passive Screens can duplicate the state of the packet filter engine should the active Screen fail. The secondary Screens have the same rules and process the packets in the same way.

Note – Both Screens mirror configuration. They attempt to mirror state by independently building the same state table, since they see the same traffic. They do not exchange information about what is in each other's state tables, however. That means that when one Screen is rebooted, it has the same rules, configuration, MAC addresses, etc., but does *not* have the same state in memory. This Screen never learns old information from the other Screen; it only learns new information from listening on the wire. The internal state as far as memory and state tables are concerned is out of sync for some undetermined amount of time, until both systems have the same state (eventually the statetables entries of the Screens that have been up longer will time-out or terminate and the statetables will be synchronous) .

The policy is stored on the primary Screen and pushed in the clear to the secondary Screens over the dedicated network connection between the primary Screen and the secondary Screens, called the HA heartbeat network.



Caution – Because the HA cluster transmits secret keys and policies in the clear over the dedicated HA heartbeat network, keep the HA heartbeat network physically secure.

To prevent the secondary Screen from sending out duplicate packets, packet transmission is automatically disabled on their filtering interfaces.

If you are administering an HA cluster remotely, there are special considerations. Because SunScreen uses the same IP addresses on all routing interfaces, it is not possible to tell from the Administration Station which Screen you are connecting to.

If the remote Administration Station connects to the IP address 172.16.3.1, the active HA Screen responds. This could be the HA primary or an HA secondary Screen. The configuration is only present on the HA primary Screen. For this reason, you must point your browser to the HA interface, which is unique, to administer the HA primary Screen:

`http://172.16.4.1:3852`

To keep the HA heartbeat network private, do not broadcast Routing Information Protocol(RIP). You must, therefore, add a static route on the remote Administration

Station—by executing the Solaris command, `route add net 172.16.2.0 172.16.3.1 1`, for example.

If you are using NAT with HA, depending on the configuration for NAT that you are using, you must add a static address resolution protocol (ARP) entry to the primary and secondary Screens in active or passive mode so that NAT works after failover. You must replicate all non-SunScreen ARP entries, including static ARP entries, on all HA Screens. Because you must do this every time an HA Screen fails over, it is easier if you edit or create your own startup script to add the static ARP entries. See Chapter 7 and the `arp(1M)` man page for more information.

Remotely Administered SunScreen HA Configuration in Routing or Stealth Mode

It is possible to have a dedicated ADMIN interface for all administrative traffic. The HA cluster can be set up in routing or stealth mode.

Note – In stealth mode, you must have a separate ADMIN interface because the filtering interfaces (type STEALTH) have no IP addresses.

You must define an interface as type ADMIN on each Screen in the cluster that you want to administer and connect these ADMIN interfaces with their own network. You must configure the ADMIN interfaces under Solaris first.

By default, the only traffic a Screen allows on an ADMIN interface is the TCP ports 3852 or 3853 that remote administration uses. You must encrypt all traffic over an ADMIN port using certificates that you have defined in the Screen's configuration. The default configuration uses a remote administration rule that allows access to the Screen from any system that has a certificate in the certificate group `admin-group`. The Screen does not check its IP address, just its certificate.

The Administration Station must have the correct SKIP ACLs (access control lists) or the `ipseccnf` correctly setup to encrypt traffic to the ADMIN interfaces using the certificate defined as `admin.screen_name`. You can check the SKIP configuration on the Administration Station with the `skiptool` GUI or by looking at the file `/etc/skip/acl.interface_name`. See the man page for `ipseccnf(1M)` for information on configuring IPsec on the Administration Station.

The certificate names are the same because the secondary Screens do not have unique certificates. They have the same certificate as the primary Screen. When a policy is activated, the primary sends its private key and public key to the secondary Screens over the HA interface along with the objects and rules that are used in the policy. This information is *not* encrypted, so the HA interfaces should connect only to other HA interfaces and should be kept secure.

Services Allowed on The HA and ADMIN Interfaces

By default, only administrative traffic (ping and SunScreen Administration services) is allowed on the HA interface. This design keeps the network as secure as possible. However, an administrator may have some need to open up other services on this private network. This can be accomplished by adding filtering rules that include the HA network as the destination address. For example, suppose that the dedicated HA network is 172.16.0.0/24. The following policy would allow Telnet traffic to and from any address on the HA network:

```
edit> list interface
qfe0 "qfe0" HA "hanetwork" INCOMPLETE
edit> list address hanetwork
"hanetwork" RANGE 172.16.0.0/24
edit> list rule 1
1 "telnet" "hanetwork" "hanetwork" ALLOW
```

Note – The destination address must be the same network object that is used in the interface definition. An equivalent object with a different name will not work. See “To Allow Non-Administrative Traffic on an HA Network” in the *SunScreen 3.2 Administration Guide* for more information.

Traffic allowed over the ADMIN interface is defined by the service Remote Administration, which by default is just TCP port 3852 or port 3853. To allow the Administration Station to Telnet to any of the Screens, add a filter to the Remote Administration service.

The traffic on the ADMIN interface must be encrypted. If it is not encrypted, the Screen drops it.



Caution – Before changing the default behavior, consider the security implications of opening up access to your firewall. Do you *really* want or need to allow access? If you decide to make changes, be sure that Administration Station is in a secure location.

Administering the Secondary Screen

You usually do not change a configuration by administering the secondary Screen. If you connect to the secondary Screen and change the configuration, you are actually editing a different policy. The policy that you are editing is usually the one created during installation to allow the primary Screen to administer the secondary Screen. If you change the configuration on the secondary Screen, the primary and secondary Screens are no longer synchronized. If you do not break the HA cluster when you change the configuration, the changed configuration will be overwritten the next time you activate a policy on the primary Screen.

Connect to the secondary Screen only to do the following:

- Perform Solaris administration, such as adding patches
- Check the HA status of the Screen
- Manually failover the Screen
- Download logs that do not exist on the primary Screen. If the primary Screen was down, the logs exist only on the secondary Screen.
- Change the policy, if the primary Screen fails and is down for an extended period.

HA Using Switches

The switch keeps a lookup table of which MAC address is attached to which physical port on the switch. It does not send packets out of the second port to a secondary Screen. This means that the hub for the HA cluster must be between the switch and the HA cluster.

Ideally, a security policy should not have a single-point of failure through which all traffic must pass. Using the same switch for both sides of the firewall, therefore, is not a good idea, even if each side of the switch is a different VLAN (virtual local area network).

You can configure some switches to work like virtual hubs. These switches work with SunScreen.

HA Network Connections and Failovers

Once the HA cluster is running, the active and passive Screens poll each other every few seconds to verify connectivity and status. If the active Screen fails or becomes unavailable, the passive Screen that has been running the longest takes over within 15 seconds. During this time (before the passive Screen takes over), no traffic goes through the HA cluster.

HA is designed to maintain the great majority of network connections. During a reboot (an orderly shutdown), the active Screen being rebooted notifies the passive Screens, and the appropriate passive Screen takes over as the active Screen without loss of connections. Because the passive Screens do not forward, reject, or log packets, the load on passive Screens is less than the load on the active Screen. Consequently, load-induced faults that affect the active Screen are unlikely to have affected the passive Screens. Once the previously-passive secondary Screen becomes active, of course, it is subject to the same load that caused the failure.

Failover can disrupt the following connections:

- Continued connections, for protocols that keep state (memory), such as TCP connections
- Stateful connections, such as FTP, NFS, NIS, and RPC

These connections can be lost under any of the following conditions:

- The Screen taking over the filtering does not have the same state information when the failover condition occurs
- A connection through the HA cluster uses dynamic NAT and two or more connections have identical destination addresses, destination ports, or source ports

HA automatically disconnects if it is only running on one system, allowing it to act like a standard Screen.

You can configure a Screen as part of an HA cluster during installation. Alternatively, you can configure HA settings through the command line, as described in Appendix B.

Configuring HA

High Availability has the following limitations:

- Screens in an HA cluster do not exchange state information. They accumulate state independently by processing the same packets. For optimal performance, whenever possible, an HA cluster should comprise systems of equal size (same CPU speed, and memory size). A slower Screen in passive mode can fall behind in processing and eventually fail to have the same state as the active Screen.

Note – To reinstate a repaired Screen, install it as a secondary Screen and add it to the cluster. You do not have to switch back to the original Screen except when its abilities or speeds are better. You can force the faster system to take over as the active Screen, or wait until both systems have the same state (eventually the statetables of the out-of-sync Screens will time-out and become synchronized) .

- Secrets are sent “in the clear” on the dedicated HA network to facilitate administrative communication within the HA cluster. Reserving the HA interface as a separate network from other traffic helps prevent discovery of the Screen’s private certificates.
- Each Screen in an HA cluster must have a unique node name and a unique IP address on the dedicated HA network. All other aspects of the Solaris system configurations must be identical on all Screens in the HA cluster. This includes the

configuration of all network interfaces (other than the HA interface).

- Proxies connect directly to the active Screen. These proxy connections will not be on the new active Screen if the previously-active Screen fails.
- HA Screens must be connected to each other through a shared network (such as a 10BaseT or 100BaseT hub) and not through a switched network (such as an Ethernet switch) because a switched network will send each packet to only one HA Screen, rather than to all of them.
- SunScreen HA in routing mode does not support FDDI, token ring, ATM, Gigabit Ethernet, or failover of IKE-based IPsec connections.
- Because passive HA Screens do not generate traffic (other than automatic administrative traffic, such as HA administration and HA heartbeat), you cannot use `telnet` for connecting to passive HA Screens. You can use `telnet` for communicating with the active HA Screen only.
- The HA cluster decides internally which member will be the active Screen. If the active Screen fails, the secondary Screen that has been running the longest becomes active.

Administering HA

If the HA cluster has an ADMIN interface, you can use the ADMIN interface's IP address to administer the Screen. The ADMIN interface must be on the primary Screen. This is the normal setup for stealth mode and is the best way to set up routing mode as well.

If the HA cluster does not have an ADMIN interface, the Administration Station needs to connect to a unique IP address to determine which Screen is the primary and which are the secondaries. The filtering interfaces share the same IP address in routing mode or have no IP address in stealth mode. The only interface with a unique IP address is the HA interface. You must connect to the HA interface of the primary Screen for administration.

The configuration information is stored only on the primary Screen. If you want to change the configuration with remote administration, you must connect to the primary Screen using an ADMIN interface or the HA interface. The primary does not have to be the active Screen. A passive primary Screen still receives and transmits administration traffic.

If the addresses for HA interfaces on the dedicated network connecting the HA Screens are unregistered, you can still administer the primary Screen. If the HA cluster and the Administration Station are both connected to the network for which the Screens are filtering traffic, the Administration Station has a route to the HA interface

of the primary Screen. They can, therefore, communicate with each other. Problems can occur when the Administration Station cannot connect directly to one of the Screen's filtering interfaces and the packets from the Administration Station must be routed to the Screen. In this case the routers in between must also know about the unregistered HA interfaces.

Authentication

SunScreen provides the ability to configure user entities. These entities are used to authenticate individual administrators *of* Screens and to allow access *through* the Screen when using proxied services. This chapter discusses the following topics:

- “User Authentication” on page 129
- “Authorized User” on page 130
- “Administrative User” on page 135
- “Proxy Users” on page 135
- “RADIUS User Authentication Details” on page 143
- “SecurID User Authentication Processing Details” on page 149

User Authentication

Authentication enables you to verify the identity of both internal and external users based on user name and a simple text password, or on a user name and SecurID token passcode, or both.

Proxies provide a means to validate, regulate, and extend the abilities of certain services beyond those afforded by kernel-based stateful packet filtering. (See Chapter 10).

User Identification

SunScreen contains two aspects of user identification: authorized user (as defined in the `authuser` database) and proxy user (as defined in the `proxyuser` database). The `authuser` database describes unique individuals; the `proxyuser` database identifies *mappings* of individuals to roles, which specify what given users can do. In addition,

there is an alias—administrative users—for authorized users. Administrative users are functionally identical to authorized users and their role is defined by the access rules.

The administration GUI edits authorized users, which are `authuser` objects; administrative users, which are `adminuser` objects; or proxy users, which are `proxyuser` objects.

Authorized User

"Authorized user" is a named common object that describes an individual user who is distinct from all others. The attributes provide a repository for demographic and authentication data about that individual.

Access to and use of the administrative GUI functions require that you establish the authorized user identity before administration is allowed. Both the administration GUI Login screen and the `login` subcommand of the `ssadm` command line reference an authorized user object.

Authorized user objects contain information sufficient to allow authentication of users of SunScreen. Validation information can either be a simple-text password or a SecurID token passcode. Users can also be configured to have both means of authentication.

The authenticity of an authorized user only establishes the identities of individual administrators, not the various roles they may play while using SunScreen. Role is established in one of two ways: (1) reference within the User field in the administrative access rules of a policy, or (2) reference from a packet filtering rule that uses user authentication (proxies).

Defining an Authorized User Object

Note – In examples, the names of authorized users, proxy users, and other user naming items are often different for purposes of clarity and illustration.

You can create and manage the authorized user and proxy user objects through the administration GUI and the command line interface. This section describes the attributes of these objects and their manipulation using the command line.

The authorized user object contains the following items:

- *name* name of the entity (1 to 255 characters)
- *enabled* | *disabled* – The flag for the entire object. If *disabled*, authentication of the associated user is always denied. The default is *enabled*.
- *password*= { *pwitem* } (optional) – A simple-text password for this user.
- *securid*= { *siditem* } (optional) – A SecurID mapping for this user.
- *real_name*= "*rnstr*" (optional) – A demographic string that can be used to identify the person in a more readable form.
- *contact_info*= "*cistr*" (optional) – A demographic string that can be used to automate contact with the person (for example, electronic mailbox address).
- *description*= "*descstr*" (optional) – A demographic string that can be used to store other notations about the person.

Note – Either a *password* item or *securid* item or both must be present for any authorized user object.

The *password*= and *securid*= items define *authentication methods* for the authorized user.

The *password*= item has the following subitems:

- *passwd* – The plaintext password string. It is either empty (" ") or it contains a one to eight-character password; if this field is not empty, then the next subitem (*crypt_password*=) does not appear.
- *crypt_password*=*cryptpasswd* (optional) – The encrypted version of the plaintext password string. If this subitem is present, the plaintext password string (above) is empty
- *enabled* | *disabled* – The flag for this simple-text password authentication method. If *disabled*, any password presented for authentication of this user is not compared against this subitem. The default is *enabled*.

The processing of *passwd* and *crypt_password*= subitems is special. When an authorized user object is first created (or whenever a new password is set for that user), the password can be presented in plaintext using the (nonempty) *passwd* subitem. Thereafter (for example, whenever the object is edited), the *crypt_passwd*= subitem can be used to retain a password without having to know (or retype) the plaintext form.

The encryption method used for these objects is identical to that used by Solaris to encrypt user passwords (those stored in */etc/shadow*). This provides the ability to *clone* encrypted passwords from Solaris to SunScreen user descriptions without the SunScreen administrator needing to know the users' plaintext passwords. This also means that the content of the SunScreen authorized user database is maintained with file permissions that prevent access from all but *root* users of the SunScreen.

The `securid=` item has the following subitems:

- `"securidname"` – User login name associated with this users' SecurID token in the ACE/Server database.
- `enabled | disabled` – The flag for this SecurID authentication method. If disabled, any password presented for authentication of this user is not submitted to the ACE/Server. The default is enabled.

Note – If both simple-text and SecurID methods exist in a single authorized user object, the simple-text method is presented first.

Creating an Authorized User Object

The authorized user object is manipulated using the `authuser` subcommand of `ssadm edit`. The `authuser` subcommand takes one of the following verbs:

- `add "name" item...` – Creates or overwrites an object; takes a complete description of the object, beginning with its name, followed by desired items and subitems as defined above.
- `delete "name"` – Deletes a named object
- `print [, sortopt] ["name"]` – Display one or more objects. If an object's name is given, then only that object's definition is displayed; otherwise all authorized user objects are displayed
- `names [, sortopt]` – Displays the names of all objects. *sortopt* can be:
 - `asc` – ascending order by name (case-sensitive)
 - `desc` – descending order by name (case-sensitive)
 - `iasc` – ascending order by name (case-insensitive)
 - `idesc` – descending order by name (case-insensitive)
 - `raw` – order stored in database

The default is `asc`.

Example: Displaying An Authorized User

The following is an example of what you type to display an existing authorized user object while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser print jeff.hogg
"jeff.hogg" ENABLED PASSWORD={ " " CRYPT_PASSWORD="s8Q2DZRw4tmGk" ENABLED }
DESCRIPTION="large and in charge" REAL_NAME="Jeff Hogg"
```

print surrounds the value of each item in double quotes. These are only necessary on input to protect embedded spaces within the values of items or to preserve null items.

print produces all tag names in capital letters (for example, REAL_NAME=). On input, the case for these tags is not important (for example, real_name= and REAL_NAME= are equivalent).

Because of the way in which passwords are encrypted, the add operation is unlikely to yield a particular crypt_password= encoding of any given plaintext password. There are 4096 different encryptions of any given plaintext password.

Examples: Creating Authorized User Objects

Following are examples of creating authorized user objects.

The following is an example of what you type to create the above authorized user object while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser add jeff.hogg password={ "4flash" }
description="large and in charge" real_name="Jeff Hogg"
edit> quit
```

This shows creation of the object by supplying the simple-text password in the plaintext form.

To create the above authorized user object, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser add jeff.hogg password={ "" crypt_password="s8Q2DZRw4tmGk" }
description="large and in charge" real_name="Jeff Hogg"
edit> quit
```

This shows creation of the object by supplying the simple-text password in its already encrypted form

In each of the above add operations, the items have been allowed to default to enabled.

To re-create the above authorized user object so that it is disabled while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser add jeff.hogg disabled password={ "" crypt_password="s8Q2DZRw4tmGk" }
description="large and in charge" real_name="Jeff Hogg"
```

To create an authorized user object defining a SecurID authentication method, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser add jeff.hogg securid={ "jeffh" }
```

```
description="large and in charge" real_name="Jeff Hogg"
```

To create an authorized user object defining both simple-text password and SecurID authentication methods, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser add jeff.hogg password={ "" crypt_password="s8Q2DZRw4tmGk" }
securid={ "jeffh" } description="large and in charge" real_name="Jeff Hogg"
```

To display all authorized user objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser print
"admin" ENABLED PASSWORD={ "" CRYPT_PASSWORD="1hp1R.xm.w63Q" ENABLED }
DESCRIPTION="(created by install) REAL_NAME="SunScreen Administrator"
"jeff.hogg" ENABLED SECURID={ "jeffh" ENABLED }
DESCRIPTION="large and in charge" REAL_NAME="Jeff Hogg"
```

To display the names of all authorized user objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> authuser names,raw
"admin"
"jeff.hogg"
```

Authorized User Authentication Processing Logic

Authentication processing is performed in the order of authentication methods in the authorized user object.

First, if the authorized user object itself is disabled, authentication fails.

Second, if the simple-text password method exists and is enabled, then the password supplied is encrypted and compared against the one stored in the method subitem. If they are equal, then authentication succeeds.

Third, if the SecurID method exists, is enabled, and the password presented appears to be a possible SecurID passcode (that is, ends in 6 decimal digits), then it is submitted to the ACE/Server along with the *securidname* for the method. If the ACE/Server indicates success, then authentication succeeds.

If none of the above yields success, then authentication fails.

Administrative User

The administrative user object identifies the SunScreen administrators that administer the Screen. The administrative user object is, in reality, a role of an authorized user.

After you create an administrative user object, you grant administrative access by creating an administrative access rule. The name that you create for the administrative user object is the same name that you use when you create administrative access rules.

Proxy Users

Proxy user is a named common object distinct from the authorized user. Proxy users are either SIMPLE or GROUP objects. SIMPLE objects are used to provide for and establish an association between individual authorized users and the role they play in using the facilities controlled by SunScreen. GROUP objects are used to allow creation of groups of SIMPLE proxy users that share common access to facilities. Thus, GROUPs streamline the task of allowing or removing access.

Some special proxy user objects also provide the means to *map* external collections of users into the access control facilities of SunScreen. In SunScreen, external access to SecurID users and RADIUS users is provided. (Access to other external user databases is afforded using RADIUS as an intermediary agent. For example, access to LDAP user databases stored through Sun Directory Services (SDS) are accessible through RADIUS.)

The figure below summarizes the relationship between rules, authorized users, proxy users, and external user databases.

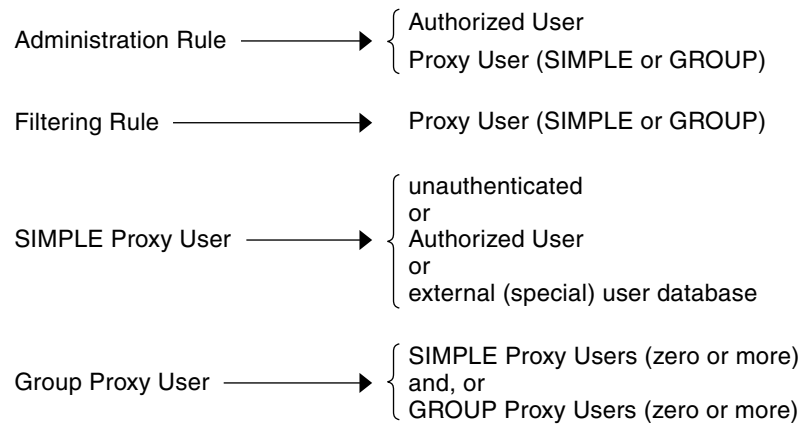


FIGURE 9-1 Summary of the Relationship Between Rules, Authorized Users, Proxy Users, and External User Databases

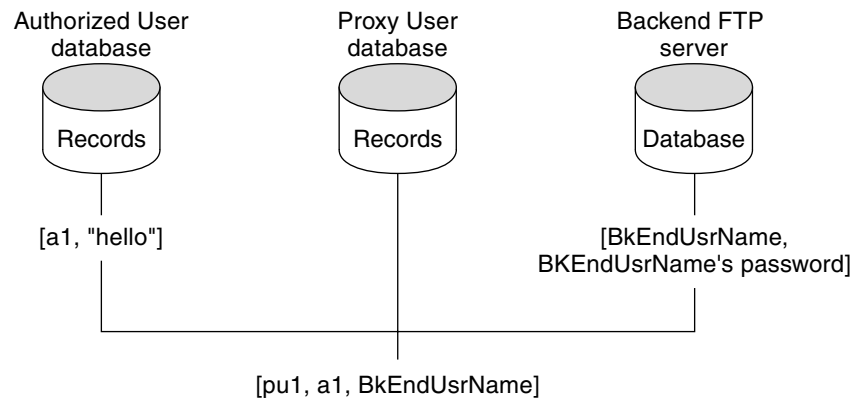
Authorized users and proxy users names are distinct, and you can have objects with identical names in each. Choose a naming strategy for each set that best reflects the naming systems already employed. For example, you can choose to name authorized users by employee identities, like distinguishing names or employee numbers, and proxy users by names that reflect their normal user login names deployed on server systems (for example: UNIX login name). Names may not contain the following characters:

! # \$ % ^ & * { } [] < > " \ ? ' / @ NULL

Names of authorized users, proxy users, and other user naming items are often deliberately chosen to be different for purposes of clarity.

The figure below shows the relationship of the proxy user to the authorized user and backend user name's password.

The records in the proxy-user database provide the link between the records in the authorized-user database and the backend user's name in the server's user database.



Example: Proxy User Name – pu1
 Authorized User Name – a1
 Authorized User Password – hello
 Backend User Name – BkEndUsrName

FIGURE 9-2 ProxyUser Example

Defining a Proxy User Object

The SIMPLE proxy user object is used to define associations between user authentication mechanisms and the identity a user assumes when connected to a permitted network resource. This association is loosely dubbed a *role*.

A SIMPLE proxy user object can indicate one of three types of authentication to be used: (1) none, (2) an authorized user object, or (3) an external authentication mechanism.

The relationship between SIMPLE proxy users and the authentication mechanism was shown in Figure 9-1.

A SIMPLE proxy user object also indicates the user identity string to be supplied when establishing the user identity on a network resource. This network resource is dubbed the backend server and, by derivation, the identity established on the backend server is defined by the *backend_user_name* item.

Note – The *backend_user_name* is only used by the FTP proxy.

A GROUP proxy user object is a collection of one or more references to other proxy user objects, either SIMPLE or GROUP.

Any proxy user object, either SIMPLE or GROUP, contains the following items:

- *name* – Name of the entity (1 to 255 characters).
- *enabled* | *disabled* – The flag for the entire object. If disabled, authentication of the associated user is always denied. The default is *enabled*.
- *group* | *simple* – The type designator of the object. You can usually omit this on input because it can be deduced from the presence of other type-specific items.
- *description="descstr"* (Optional) – A demographic string that can be used to store notations about the role.

A SIMPLE proxy user object contains the following items:

- *radius* | *securid* (optional) – Indicates that this object is a *SPECIAL* one, associated with unrestricted mapping of users from the RADIUS or SecurID system (an external authentication method). Only one *SPECIAL* indicator can be present in a given proxy user object. If present, the next (*auth_user_name=*) item should not be given.
- *auth_user_name="auser"* (optional) – Indicates that the name of an authorized user object that is used to authenticate this user role. If it is absent, and if no *SPECIAL* item is present, then the proxy user object requires no authentication.
- *backend_user_name="beuser"* – Gives the backend user name string to supply when establishing the user's identity on a backend server. If no *SPECIAL* item is present, then this item is required, otherwise; it is ignored.

A GROUP proxy user object contains zero or more of the following items:

- *member_name="memname"* – Gives the name of another proxy user object that is a group member.

Note – Although you can add a GROUP proxy user object, including a complete list of its members, the special commands *addmember* and *deletemember* are provided to edit the membership list of a GROUP.

Creating a Proxy User Object

The proxy user object is manipulated using the *proxyuser* subcommand of *ssadm*. *edit.proxyuser* takes one of the following as commands:

- *add "name" item . . .* – Creates or overwrites an object. It takes a complete (perhaps initial, in the case of GROUP) description of the object, beginning with its name, followed by desired items, as defined above.
- *delete "name"* – Deletes a named object.
- *addmember "grpname" "memname"* – Adds a member to an existing GROUP object. Duplicate *addmember* operations are ignored.

- `deletemember "grpname" "memname"` – Deletes a member from an existing GROUP object. Any attempts to remove an unknown member are ignored.
- `print [,sortopt] ["name"]` – Displays one or more objects. If an object name is given, then only that object's definition is displayed; otherwise, all proxy user objects are displayed.
- `names [,sortopt]` – Displays the names of all objects. *sortopt* can be:
 - `asc` – ascending order by name (case-sensitive)
 - `desc` – descending order by name (case-sensitive)
 - `iasc` – ascending order by name (case-insensitive)
 - `idesc` – descending order by name (case-insensitive)
 - `raw` – order stored in database

The default is `asc`.

Examples: Displaying, Creating, Adding, Removing Proxy User Objects

Following are examples of displaying, creating, adding, and removing proxy user objects:

To display existing proxy user objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> proxyuser print jdh
"jdh" ENABLED SIMPLE AUTH_USER_NAME="jeff.hogg" BACKEND_USER_NAME="jeffh"
DESCRIPTION="Jeff Hogg as self on Solaris"
edit> proxyuser print proxyusers
"proxyusers" ENABLED GROUP MEMBER_NAME="radius" MEMBER_NAME="jdh"
DESCRIPTION="users allowed through FTP and telnet proxies"
```

To create the above SIMPLE proxy user object, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> proxyuser add jdh auth_user_name=jeff.hogg
backend_user_name=jeffh description="Jeff Hogg as self on Solaris"
edit> quit
```

To create the above GROUP proxy user object, while logged into the primary Screen. First create the initial group with no members:

```
admin% ssadm -r primary edit Initial
edit> proxyuser add proxyusers group description="users allowed through FTP
and telnet proxies"
```

The above empty group creation demonstrates a case where the GROUP type cannot be deduced from the other tags, since `description=` is a tag common to all proxy user object types.

To add the members of the example GROUP:

```
edit> proxyuser addmember proxyusers radius
edit> proxyuser addmember proxyusers jdh
```

Member names are stored in the order in which you add them to GROUP objects. The order is unimportant to authentication processing. This example also uses a SPECIAL object radius that is defined during initial installation.

In each of the above add operations, the items have been allowed to default to enabled.

Following are examples of displaying and removing proxy user objects.

To remove a member reference from a GROUP proxy user object, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> proxyuser deletemember proxyusers radius
edit> proxyuser print proxyusers
"proxyusers" ENABLED GROUP MEMBER_NAME="jdh"
DESCRIPTION="users allowed through FTP and telnet proxies"
```

To display all proxy user objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> proxyuser print
"admin" ENABLED SIMPLE AUTH_USER_NAME="admin" DESCRIPTION="initial SunScreen administrator"
"admin-group" ENABLED GROUP MEMBER_NAME="admin" DESCRIPTION="SunScreen administrators"
"anonymous" ENABLED SIMPLE BACKEND_USER_NAME="anonymous"
DESCRIPTION="unauthenticated user, for anonymous FTP, etc."
"ftp" ENABLED SIMPLE BACKEND_USER_NAME="anonymous"
DESCRIPTION="unauthenticated user, for anonymous FTP, etc."
"jdh" ENABLED SIMPLE AUTH_USER_NAME="jeff.hogg" BACKEND_USER_NAME="jeffh"
DESCRIPTION="Jeff Hogg as self on Solaris"
"proxyusers" ENABLED GROUP MEMBER_NAME="radius" MEMBER_NAME="jdh"
DESCRIPTION="users allowed through FTP and telnet proxies"
"radius" ENABLED SIMPLE RADIUS DESCRIPTION="default, external, non-specific RADIUS
proxy_user" "securid" ENABLED SIMPLE SECURID DESCRIPTION="default, external,
non-specific SecurID proxy_user"
```

To display the names of all proxy user objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> proxyuser names,raw
"admin"
"admin-group"
"anonymous"
"ftp"
"radius"
"securid"
"jdh"
"proxyusers"
```

Proxy User Processing Logic

In earlier sections, the relationship between authorized user objects and proxy user objects was described, as well as an introduction of external (SPECIAL) proxy user authentication methods. This section gives more specifics regarding these objects and mechanisms, and the sequence of steps performed upon them by the processing logic.

There are fundamentally two types of authentication processing performed in SunScreen. Whenever a user identity is required, authentication processing occurs first, using one of these two processes. After SUCCESS is obtained in the authentication of an identity you supplied, access control matching determines if the user is allowed to perform the operation being attempted; authentication FAILURE denies any requested operation. (Access control is described in the section “User Access Control Processing Logic” on page 143.)

The first type of processing is the authentication of an authorized user object directly. This form occurs when authenticating an administrator of SunScreen, either through the administration GUI Login page or by the `ssadm login` subcommand. In these situations, the user name provided must match an ENABLED authorized user object. Authentication logic has been previously described (see “Authorized User Authentication Processing Logic” on page 134).

The second type of processing is the authentication of a proxy user object. This form occurs when authenticating a user who desires access through the FTP or telnet proxies.

As previously introduced, authentication of a proxy user object can take one of three subpaths: (1) null authentication, (2) authorized user authentication, or (3) SPECIAL external authentication method processing.

Regardless of the proxy user authentication path, only ENABLED objects can be used for authentication. A DISABLED object always results in an authentication FAILURE if attempted.

Null Authentication

Null authentication occurs when a SIMPLE proxy user object named by the user contains no `auth_user_name=` item. In this situation, whatever input is given for the password is accepted without any checking. The preinstalled `anonymous` and `ftp` proxy user objects are examples of this type of authentication.

There is nothing special about the names `anonymous` and `ftp` for these preinstalled objects. You can create additional null authentication proxy user objects and use them instead of or in addition to the preinstalled ones. This allows the creation of null authentication paths that are private to your installation.

Referenced Authorized User Authentication

Authorized user authentication occurs when an `auth_user_name=` item is present. In this situation, authorized user authentication logic is performed on the authorized user object named in the item (see “Authorized User Authentication Processing Logic” on page 134).

SPECIAL External Method Authentication

SPECIAL external authentication method processing occurs when you supply an identity, which is a compound name consisting of a SPECIAL external authentication method and a backend user name.

The syntax of this compound name is:

`/ext_method_name/back_end_username`

For example, compound names that use the preinstalled RADIUS and SecurID SPECIAL methods might be:

`/radius/jeffh` `/securid/jeffh`

There is nothing special about the names `radius` and `securid` in these preinstalled SPECIAL objects; they are distinguished by their special `radius` or `securid` items. You can create additional SPECIAL authentication methods and use them instead of or in addition to the preinstalled ones. This enables you to create authentication paths that are private to your installation, perhaps to hide these paths or to abbreviate user input.

SPECIAL external authentication logic varies depending upon the method in question. More specifics about the two external methods (RADIUS and SecurID) can be found in later sections. There are two means for using SecurID tokens for authentication: one within the authorized user object, the other through SPECIAL external proxy user objects. The reason for this apparent redundancy lies in the level of *trustedness* of the two mechanisms. When using the authorized user path, an association is formed between a specific SunScreen authorized user object and a specific SecurID token holder. The SPECIAL external authentication mechanism allows, in essence, SecurID to authenticate any *user* who passes the challenges. The ability to establish authenticity for purposes of SunScreen administration is never available to SPECIAL external authentication.

You choose the mechanism to employ depending upon the security requirements of your site.

User Access Control Processing Logic

As previously referenced, once an identity is supplied, the would-be user is authenticated. This (now authentic) user must be allowed to perform the operation presently being attempted. The logic that determines the abilities of a user is loosely termed *access control*.

As with the initial nature of user authentication, there are two contexts within which access control is employed: SunScreen administrative activities and usage of network facilities controlled by FTP, Telnet, and (optionally) HTTP proxies.

Once an administrative user has been authenticated, access to administrative capabilities is controlled by the administrative rules for both remote and local use. Within each such rule, a user name is found. If the authorized user named within the rule matches the authenticated identity, then access is granted at the level the rule specifies. Alternatively, the rule can reference a proxy user object; if the authenticated identity is a member of that proxy user object, then the associated rule-specified access is likewise allowed.

SunScreen proxies that perform user authentication do so by requiring their rules to reference a proxy user object. Once the user-supplied identity has been authenticated, that identity (which is always a proxy user in this context) is evaluated to see if it is a member of the proxy user object that the rule referenced. If it is, then the associated rule matches. (See Chapter 10.)

RADIUS User Authentication Details

The RADIUS protocol provides the ability to centralize user authentication databases for widespread uniform deployment throughout organizations. Many forms of external authentication processing, both proprietary and standard, provide the ability to configure a RADIUS agent, or *gateway*.

Among noteworthy mechanisms so endowed are:

- RSA Security ACE/Server Version 3.2 and above
- Sun Directory Services (SDS)

The SDS gateway facility enables SunScreen to tap into authentication using LDAP user databases.

With respect to a RADIUS server, the Screen plays the role of a RADIUS Requestor.

Note – The term *client* becomes confused with clients of SunScreen-*provided* services so the term *requestor* is used. It also describes the relationship between the RADIUS server and the Screen more accurately.

The RADIUS protocol uses UDP datagrams to perform user authentication. Each RADIUS Requestor is configured with a *node secret* that is known only to itself and the RADIUS servers from which it expects authentication. That node secret is used to encrypt requests and to decrypt and authenticate server responses.

The RADIUS protocol in SunScreen comes installed with nearly all parameters prefigured for immediate use. Four remaining configuration elements needing postinstallation attention are:

- Define address object for the RADIUS servers
- Define access rules allowing RADIUS protocol access to those servers
- Configure the RADIUS Requestor to use the defined server address object or objects
- Configure the RADIUS Requestor with SunScreen's RADIUSNodeSecret

The SunScreen RADIUS Requestor can use up to eight different IP addresses for servers to be queried. These can be configured in one or more address common objects with arbitrary names.

Tip – Define a single address group object to collect all RADIUS servers for ease in creating server access rules.

To enable the RADIUS Requestor to function, the Screen must be configured to allow it access to the servers through the `radius` service common object, which comes preinstalled.

RADIUS Server Configuration

The RADIUS Requestor learns of its RADIUS servers and node secret from the variables `RADIUSServers` and `RADIUSNodeSecret`, respectively.

The `RADIUSServers` variable can either be global or Screen-specific.

It contains the following items:

- `sys=Screen` (optional)
- `prg=auth`
- `name=RADIUSServers`

- `values={ host=server ... }` —Address object names or IP addresses or both of one or more RADIUS servers
- `description="descriptive text"` (optional)
- `enabled | disabled` — The default is enabled.

For multiple-Screen installations, there are at least two approaches for dealing with the possible need to have Screens use different RADIUS servers. One is to employ the SCREEN attribute on address objects with the same name, and then use a `globalRADIUSServers` variable. Another is to avoid the use of SCREEN attributes on address objects and instead use the `sys=` item to create Screen-specific `RADIUSServers` variables. Of course, combinations are also possible. Naturally, the logic prefers Screen-specific address objects and variables over global ones.

The address object or objects (referenced by *server* name in the above) can be GROUP, RANGE, or SINGLE. *server* can also be a dotted-quad IP address; however, avoid such usage unless required. The first eight unique IP addresses produced during processing of the variable are used.

Note – Because of the way SunScreen represents address objects, use of GROUP or RANGE objects results in *server* usage that is ordered by ascending IP address. The preference order of *server* use can be controlled precisely by the order of the subitems in the `values={ ... }` of the `RADIUSServers` variable.

RADIUS Node Secret Configuration

The `RADIUSNodeSecret` variable specifies a character string to use for security and authenticity when interacting with the configured RADIUS server or servers. Because of the way RADIUS operates, only the RADIUS requestors have node secrets (not the servers).

The same value configured for `RADIUSNodeSecret` must also be introduced into each RADIUS server through its own configuration mechanism. (For obvious reasons, this should be done in an out-of-band fashion.)

The `RADIUSNodeSecret` variable is normally Screen-specific. It contains the following items:

- `sys=screen` (optional)
- `prg=auth`
- `name=RADIUSNodeSecret`
- `value="nodesecret"`
- `description="descriptive text"` (optional)
- `enabled | disabled` — The default is enabled.

In multiple-Screen installations, the `sys=` item enables you to configure different node secrets for each Screen.



Caution – Because shortcuts were taken by some reference implementations, a common deficiency in RADIUS servers is the handling of node secrets that are longer than 31 characters. If you intend to use longer values, first determine that your server or servers can handle them correctly.

Once you establish addresses, rules, and variables, you must activate the configuration to propagate the changes.

Typical RADIUS Configuration

A typical RADIUS configuration scenario has two Screens that each protect a site. `la-screen` and `la-radsvr` are a Screen and RADIUS server in the `la` location, `sf-screen` and `sf-radsvr` are a Screen and RADIUS server in the `sf` location. Each site uses the RADIUS server of the other as a backup.

Note – Ephemeral IP addresses are shown. Encrypted tunnels, or VPNs, are possible, perhaps likely, in such a configuration, but are not shown for purposes of clarity.

Examples: Typical RADIUS Configurations

The following are examples of typical RADIUS configurations:

To create address objects, while logged into the primary Screen:

```
admin% ssadm -r primary edit ConfigName
edit> add address la-radsvr HOST 1.2.3.4 ...
edit> add address sf-radsvr HOST 4.3.2.1 ...
edit> add address radsvrs GROUP { la-radsvr sf-radsvr } { } ...
```

To create a rule to allow RADIUS Requestor-to-server access, while logged into the primary Screen:

```
edit> add rule radius localhost radsvrs ALLOW
```

To create RADIUS variables, while logged into the primary Screen:

```
edit> vars add sys=la-screen prg=auth name=RADIUSServers
values={ host=la-radsvr host=sf-radsvr } description="RADIUS servers for la site"
edit> vars add sys=sf-screen prg=auth name=RADIUSServers
values={ host=sf-radsvr host=la-radsvr } description="RADIUS servers for sf site"
```

To create RADIUS node secret variables, while logged into the primary Screen:

```
edit> vars add sys=la-screen prg=auth name=RADIUSNodeSecret value=la-secret
edit> vars add sys=sf-screen prg=auth name=RADIUSNodeSecret value=sf--secret
```

To save and activate the configuration:

```
edit> save
edit> quit
admin% ssadm -r primary activate ConfigName
```

For example, given a valid, RADIUS-hosted user gooduser with password goodpass and an invalid user baduser, while logged into the Screen la-screen:

```
admin% ssadm -r la-screen lib/user_authenticate -v /radius/gooduser goodpass
User /radius/gooduser authenticated and mapped to backend user gooduser
admin% ssadm -r la-screen lib/user_authenticate -v /radius/gooduser anythingelse
User /radius/gooduser failed authentication.
admin% ssadm -r la-screen lib/user_authenticate -v /radius/baduser anything
User /radius/baduser failed authentication.
```

Other vars for RADIUS Configuration

The following additional variables are preinstalled and used to control the RADIUS client protocol; they are pre-ENABLED and generally need not be altered.

The number of seconds that the requestor logic waits before contacting an unresponsive server:

```
edit> vars print PRG=auth
PRG="auth" NAME="RADIUSHolddown" ENABLED VALUE="300"
DESCRIPTION="seconds to ignore a non-responsive RADIUS server"
```

The number of passes through the server list that requestor logic makes before giving up:

```
PRG="auth" NAME="RADIUSRetryPasses" ENABLED VALUE="3"
DESCRIPTION="how many times to try each RADIUS server"
```

The name of the RADIUS server port, as given in the Service registry:

```
PRG="auth" NAME="RADIUSService" ENABLED VALUE="radius"
DESCRIPTION="RADIUS service / port # at which to query server(s)"
```

The number of seconds for each response that requestor logic waits before sending another attempt:

```
PRG="auth" NAME="RADIUSTimeout" ENABLED VALUE="5"
DESCRIPTION="seconds to await each RADIUS server response"
```

The requestor logic attempts to contact only servers that have not been held down during the first pass; subsequent passes contact each server regardless of previous nonresponsiveness. During the first pass, each server is contacted twice in a row

before moving onto the next one. During subsequent passes, each server is only contacted once. A rough upper-bound on the overall time for total failure for all servers is:

$$\text{\#servers} \times (\text{\#passes} + 1) \times \text{timeout}$$

This is an upper-bound because of the way the first pass avoids recently unresponsive servers; a lower-bound would be:

$$\text{\#servers} \times (\text{\#passes} - 1) \times \text{timeout}$$

So, for example, with two servers configured and using the default time-outs, the overall failure time-out would be less than $2 \times (3 + 1) \times 5 = 40$ seconds, and greater than $2 \times (3 - 1) \times 5 = 20$ seconds

Other RADIUS Protocol Notes

The requestor implementation only attempts to use authentication. It does not ask the server to store any accounting information. It allows for a single node secret for each Screen. Theoretically you could use a distinct node secret for each server. This approach was not used. Rather than complicate configuration by allowing any given Screen to have multiple node secrets (one for each server), a simpler, easier to configure approach was chosen.

RADIUS Testing

The requestor implementation has been tested against two commercial server implementations provided with BSDI's BSD/OS. The first version tested was included in BSD/OS Version 3.1, and is derived from the Livingston reference implementation. The second version is as delivered in BSD/OS Version 4.x; this version is based on the University of Michigan Merit AAA RADIUS server implementation.

Testing is done against Sun Directory Services (SDS) and ACE/Server 3.2 SecurID authentication.

RADIUS Usage

As discussed in the earlier part of this chapter, the RADIUS authentication mechanism is used within SunScreen in the form of a (SPECIAL) External Method. (See "SPECIAL External Method Authentication" on page 142.)

You can use RADIUS authentication of a user only within the proxy context in SunScreen. You *cannot* use RADIUS authentication for authenticating an administrative user.

RADIUS users are permitted by connecting a SPECIAL proxyuser entity with one or more proxy policy rules. This connection can be directly in the rule or by inclusion in a group used in the rule. (See “Proxy Users” on page 135.)

SecurID User Authentication Processing Details

SecurID is a one-time password mechanism supplied by RSA Security. SecurID is a leading form of hardware-based authentication.

SecurID authentication involves three components: a user-held hardware device (token), client software that solicits input from the token-holding user, and server software that verifies the user authentication information supplied by the token-holder through the client software. The client software runs on a variety of standard operating system platforms (those capable of providing user-level security) and other imbedded system applications. The server software runs on a more restricted set of standard operating systems.

The client software portion (when installed on the Solaris operating system) is known by two names: ACE/Client and ACE/Agent. Versions of the RSA Security offering before v3.2 used the former name; version 3.2 and later use the latter name (the renaming reflects an extension of functionality). Regardless of version, the server component is known as ACE/Server. ACE/Client or ACE/Agent software from any version 3.x can properly communicate with any ACE/Server version 3.x system with a version greater than or equal to it (the client version is less than or equal to the server version).

SunScreen is compatible with ACE/Server version 3.0.1 and greater.

The SunScreen product *does not* include the ACE/Server product, which must be purchased separately.

Typical SecurID authentication involves a hardware device (token) that generates a pseudorandom value. That value is combined with a personal identification number (or PIN) to realize a *two-factor* authentication scheme. The algorithmic data for computing the pseudorandom value as well as a user’s PIN are known only to the token-holder and the ACE/Server. There are several styles of SecurID token device as well as a software implementation, but all operate in basically the same fashion.

In interfacing SecurID to SunScreen, you are expected to understand the ACE/Agent and ACE/Server implementation to a level sufficient to install and configure the SunScreen system as a client of ACE/Server. Further details of the complete SecurID facility, token types, options, and so forth, should be referred to your ACE/Server administrator.

ACE/Client, ACE/Agent, and the SunScreen Stub Client

Configuration of SecurID for use within SunScreen is relatively complex. This complexity is due partly to the third-party nature of SecurID, partly to a larger selection of options for its setup and use, and partly because the authentication provided by SecurID is robust.

The first several sections following this one discuss the various components and choices for setup that are possible. Those sections are not written in a task-oriented fashion; rather, they attempt to bridge the gap in understanding SunScreen mechanisms and the third-party offerings of SecurID.

The section “Typical SecurID Configuration” on page 154 presents an almost maximal configuration. The setup and initial testing of SecurID depends on following the applicable steps in that section.

The RSA Security ACE/Agent software offering is only supported on SPARC versions of the Solaris software through version 2.6 (SunOS 5.6). Yet, SunScreen is supported on Solaris 2.6 software and beyond, and on both SPARC and the Intel platforms. To complete the SunScreen support matrix, Sun has developed a stub client installation mechanism.

The stub client allows SunScreen to be configured with a *minimum* of information such that it can communicate with an ACE/Server for purposes of authenticating users of SunScreen-protected resources. The stub client *does not* provide the full suite of functions available within the ACE/Agent, *nor* does it supplant the need to purchase and deploy the ACE/Server software and SecurID tokens from RSA Security.

In summary, for SecurID support for SunScreen, if you are installing SunScreen on a SPARC-based Solaris machine which has a supported ACE/Agent implementation, you can choose either the stub client or the complete ACE/Agent installation on the Screen. For SunScreen on other Solaris platforms or versions, you *must* use the stub client.

SecurID ACE/Agent

The installation of ACE/Agent can be performed before or after the installation of SunScreen. The SecurID stub client configuration step can be performed any time after SunScreen installation. SunScreen does not require SecurID to function, so you can perform basic installation and configuration of the Screen first and, once running, add SecurID authentication as needed before full-scale deployment.

For purposes of SunScreen and its usage of SecurID authentication, the SecurID client software must be installed on any Screens that use SecurID authentication. For example, if only users of proxies are authenticated using SecurID, then the client software need only be installed on Screens that run proxy servers. If SecurID is used for authentication of SunScreen administrators, then the client software must be installed on all Screens. You do not have to install SecurID software on the SunScreen Administration Station platform (for remote administration), or on the end-systems of users of SunScreen-protected resources (for example, proxy clients or backend servers).

For information on installing ACE/Agent, see the documentation for that product. One important note regarding ACE/Agent use on SunScreen is that you do *not* have to actually *create* Solaris user accounts on the Screens that are protected by ACE/Agent login mechanisms to enable the authentication of SunScreen users by that Screen. (You should use ACE/Agent authentication to secure the Solaris platform of a SunScreen system in any way deemed important for administration of that system as a Solaris platform; but you do not have to make any changes to the Solaris user configuration to use SecurID fully within SunScreen itself.)

With those notes, all other issues regarding use of SecurID within SunScreen are common to both types of client software installation. The following section discusses the stub client.

SecurID Stub Client

Two files required for the SecurID stub client are loaded onto the Solaris system when the SunScreen packages are added. They are:

- `/usr/lib/sunscreen/lib/securid_stubclient_setup`
- `/usr/lib/sunscreen/lib/securid_stubclient.tar`

In addition, the stub client installation requires a file called `sdconf.rec` that is created on the ACE/Server.

The instructions for creating this file are found in the ACE/Server documentation and your ACE/Server administrator must provide this file. `sdconf.rec` contains addressing information for your ACE/Servers (master and slave) as well as cryptographic data that enables the SecurID client to establish secure and authentic communication with the ACE/Server.

When ACE/Server administrators create `sdconf.rec`, they must first inform the server of the SunScreen system. The ACE/Server must consider the Screen to be a client (specifically, a UNIX client system). The ACE/Server must also be configured to know the IP addresses of the Screen. *All* of the IP addresses that the Screen will use to access the ACE/Server must be configured into the server.

Once the above configuration is performed on the ACE/Server and is saved, the `sdconf.rec` file contains the information needed to run the stub client installation. You must get the `sdconf.rec` file from your ACE/Server administrator and onto the system.

To complete the stub client installation, you must be *root*.

Change into the directory where you loaded `sdconf.rec` and execute the setup script by typing:

```
# /usr/lib/sunscreen/lib/securid_stubclient_setup sdconf.rec
```

The script creates and deposits a few files into the `/opt/ace` directory and creates the `/etc/sdace.txt` file. It also edits `/etc/inet/services` to add a pair of service definitions required by SecurID.

SecurID Access Paths

The protocol used to communicate between the SecurID client and its servers is based on UDP datagrams. This protocol typically uses port 5500, although this can be altered by changing the configuration on *all* client and ACE/Server systems. Awareness of this port number assignment on SunScreen is found in two places:

- `/etc/inet/services`
- Registry service object in the active configuration

The protocol is named `securid` in both locations.

In more robust SecurID configurations, a slave ACE/Server is configured to serve as a backup to the master server.

As was previously described, the SecurID client software obtains the IP addresses of the ACE/Servers from the `sdconf.rec` file during client installation. Additionally, the SunScreen policy must contain rules that allow the SecurID client software on the SunScreen to access the master and slave ACE/Servers. This involves:

- Address object definition for the ACE/Servers
- Rules to allow the `securid` protocol access to the ACE/Servers

Tip – Define a single Address Group so that it collects both master and slave ACE/Servers for ease in creating server access rules.

For the master and slave ACE/Server mode to function, all SecurID clients must be able to access both servers. Additionally, the servers communicate between themselves, using an additional path through a TCP connection (typically on port 5510). You can alter this by changing the configuration on both ACE/Servers. Awareness of this port number assignment is found in the same places as that of the UDP datagram `securid` service. This TCP server-to-server protocol is named `securidprop` in both locations.

Rules are needed to allow the master and slave servers to communicate using `securidprop`.

SecurID PIN Establishment

Part of the use of SecurID tokens involves the establishment of the personal identification number (PIN). A number of variations are possible in establishing a PIN; these are all determined by the choice of SecurID token device and ACE/Server administration policy regarding PIN formulation and mode of establishment.

ACE/Server administrative choice makes it possible for the token-holders to establish their own PIN. The experienced SecurID user knows that the standard ACE/Agent client software allows establishing a token-holder PIN using the shell surrogate program `sdshell`. SunScreen does not require the use of the shell surrogate to use SecurID authentication. This approach avoids the severe security problems and administrative difficulties that are associated with creating user accounts on the Screen for each token-holder. Token-holders must nevertheless be able to establish their PIN.

The SunScreen solution is to provide a daemon process, called the PIN server. This server is started automatically whenever a policy is activated *if* the Screen has been configured as a SecurID client (either through ACE/Agent or stub client installation). The PIN server normally listens on TCP port 3855 (in the standard installation). This port number assignment is found in:

- `/etc/inet/services`
- Registry service object in the active configuration
- `/etc/init.d/proxy` startup script

In `/etc/inet/services`, it is named `securidpin`; in the active configuration, it is named SecurID PIN. In the proxy startup script, it is referenced by numeric value.

SecurID token-holders use the PIN server to establish a new PIN as necessary. Access to this server is obtained using a standard telnet client program, specifying the alternative port number (3855). For example, using the Solaris `telnet` program:

```
% telnet Screen 3855
Trying 1.2.3.4...
Connected to Screen.
Escape character is '^]'.
SunScreen V3.2 SecurID PIN / Re-keying Server
Enter SecurID login: loginname
Enter PASSCODE: passcode
```

The interaction is familiar to users of the `sds` shell and to ACE/Server administrators. Beyond the Enter PASSCODE: prompt, interaction varies depending upon the state of the SecurID token and the PIN options configured for that token on the ACE/Server.

An administrative task that must be performed on the Screen is the addition of policy rules to allow connections to the PIN server from hosts where you think allowing the PIN establishment is appropriate. For example, you may wish to require PIN establishment only from hosts *behind* your Screens and from external hosts whose traffic is protected by SKIP encryption.

Note – Some SecurID installations may not allow token-holders to do PIN establishment, opting instead for use of PINs that are determined solely by the ACE/Server administrator. In such cases, access to the PIN server is not needed.

Typical SecurID Configuration

This section attempts to bring together the various configuration elements described in previous sections with an example setup that illustrates the pertinent details of establishing a working SunScreen policy utilizing SecurID authentication.

The example presumes the following preexistent state:

- `screen` is the SunScreen Screen (as well as `localhost`)
- `admin` is the (remote) SunScreen Administration Station
- A standard Initial policy has been created, with default names for addresses and SKIP certificates
- Address objects `inside` and `outside` have been created to declare hosts that are within and without the protection of the Screen, respectively
- ACE/Servers `acemaster` and `aceslave` have been configured
- `screen` has been configured as a UNIX client in the ACE/Servers
- The (resulting) `sdconf.rec` file has been loaded into the `/var/tmp` directory on `screen`

A standard (non-PINPAD) SecurID token is used, which has been given a login name of `ssadmin`. That login has been activated on `screen` on the ACE/Servers. The token has been configured for user establishment of a 4 to 8-digit PIN and is in new-PIN mode.

The overall steps performed are:

- Stub client configuration
- ACE/Server address object creation
- SecurID client-to-server policy rule creation
- ACE/Server server-to-server policy rule creation
- PIN server policy rule creation
- Augment the SunScreen administrative user to use SecurID
- Altered policy activation
- PIN establishment
- Screen administrative authentication through SecurID

The command-line interface (using `ssadm` commands) is shown here for brevity; however, except for the stub client configuration, all other steps can be performed using equivalent administration GUI operations.

Examples: SecurID Configurations

The following are example of SecureID configurations.

To configure a SecurID stub client (while `root` in a shell on `screen`):

```
# cd /var/tmp
# /usr/lib/sunscreen/lib/securid_stubclient_setup sdconf.rec
```

To create the registry address objects to describe the ACE/Servers, while logged into the Screen:

```
admin% ssadm -r screen edit Initial
edit> add address acemaster HOST ....
edit> add address acelave HOST ....
edit> add address acservers GROUP { acemaster acelave } { } ...
edit> save
```

To continue adding the SecurID client-to-server policy rule:

```
edit> add rule securid localhost acservers ALLOW
```

To add the ACE/Server server-to-server policy rule:

```
edit> add rule securidprop acservers acservers ALLOW
```

To add two PIN server policy rules — one that allows the end-user SKIP Administration Station to access the PIN server, the other for unencrypted access for inside hosts:

```
edit> add rule "SecurID PIN" admin localhost SKIP_VERSION_2
remote screen.admin DES-CBC RC4-40 MD5 NONE ALLOW
edit> add rule "SecurID PIN" inside localhost ALLOW
```

You should place these rules early enough in the policy so that their action takes place before the action of other conflicting (DENY or less-secure) rules.

To augment the standard admin user to allow SecurID authentication (the existing value is first displayed for clarity):

```
edit> authuser print admin
"admin" ENABLED PASSWORD={ "" CRYPT_PASSWORD="1hp1R.xm.w63Q" ENABLED }
DESCRIPTION="(created by install)" REAL_NAME="SunScreen Administrator"
edit> authuser add admin password={ "" crypt_password="1hp1R.xm.w63Q" }
securid={ ssadmin } description="updated for either simple password or SecurID"
real_name="SunScreen Administrator"
```

To save and activate the augmented policy:

```
edit> save
edit> quit
admin% ssadm -r screen activate Initial
```

To perform PIN establishment of the token (from the Administration Station):

```
admin% telnet screen 3855
Trying 1.2.3.4...
Connected to screen.
Escape character is '^]'.
SunScreen V3.2 SecurID PIN / Re-keying Server
Enter SecurID login: ssadmin
Enter PASSCODE: 6-digit-passcode-from-token
New PIN required; do you wish to continue? (y/n) [n]: y
Now enter your new PIN, containing 4 to 8 digits, or press
Return to generate a new PIN and display it on the Screen, or
end the connection to cancel the New PIN procedure: 4-digit-PIN
Please reenter new PIN: 4-digit-PIN
Wait for the code on your token to change, then connect again
with the new PIN
Connection closed by foreign host.
```

The configuration is now complete. After the code on the token changes (up to one minute later), administrative access to the Screen can be obtained using SecurID. The SunScreen administrative user's name is still admin, but you supply as the password the 4-digit-PIN value (established above) followed immediately by the 6-digit value displayed by the token.

In the example, the simple-text password can also be allowed to establish administrator authenticity.

Other SecurID Details

Use of SecurID authentication by SunScreen requires the UDP and TCP protocols and also that the Screen has at least one IP address. This implies that a Screen configured with *no* routing-mode interfaces cannot use SecurID authentication (as it lacks the ability to speak these protocols).

You should exercise caution when deploying SecurID authentication to protect SunScreen administration (and indeed any other critical SunScreen-control facility where authentication is required). Because the ability to authenticate using SecurID requires using policy rules in SunScreen, a mistake in configuring a policy can leave the Screen in a state where SecurID authentication is broken. Additionally, the ACE/Server could be down or inaccessible for other reasons, which can result in an administrative lockout. As a precautionary measure, at least one SunScreen administrator should always be configured with ALL access and a simple-text password (perhaps in addition to SecurID).

ACE/Server version 3.2 and newer can be hosted on some versions of SPARC-based Solaris. Because of the additional security features of SunScreen, you may be tempted to install ACE/Server *on* the SunScreen. This is a perfectly acceptable deployment, but you are cautioned to understand *thoroughly* the ramifications of installing ACE/Server on a multihomed host. There are numerous complexities to be dealt with on an ongoing basis if your SunScreen does not have a single IP address that can service all queries from other SecurID software components. (See your ACE/Server installation documentation for information about "Multiple Server IP Addresses.")

SecurID Usage

As discussed in earlier portions of this chapter, the SecurID authentication mechanism is used within SunScreen in the form of either an authorized user or a (SPECIAL) External Method. (See "Authorized User" on page 130 and "SPECIAL External Method Authentication" on page 142.)

You can use SecurID to authenticate administrative users as well as to authenticate authorized users for SunScreen proxies.

Use of SecurID in the authorized user context is provided by configuring `authuser` entities to include SecurID references (for example, login names known to the ACE/Server or Servers).

Use of SecurID in the (SPECIAL) External Method is permitted by connecting a SPECIAL `proxyuser` entity with one or more proxy policy rules. This connection can be directly in the rule or rules or by inclusion in a group used in the rule. (See "Proxy Users" on page 135.)

Proxies

A proxy is a user-level application that runs on the Screen. The main purpose of proxies is to provide content filtering and user authentication. This chapter discusses the proxies used in the SunScreen application. It contains information on:

- “How Proxies Work” on page 160
- “Proxy User Authentication” on page 162
- “FTP Proxy” on page 163
- “HTTP Proxy” on page 166
- “SMTP Proxy” on page 173
- “Telnet Proxy” on page 179
- “VirusWall Content Scanning” on page 181

SunScreen Proxies

SunScreen enables you to set up proxies for FTP, HTTP, SMTP, and Telnet traffic protocols. Although each proxy has different filtering capabilities and requirements, you can allow or deny sessions based on source or destination addresses of packets. Proxies share common objects and policy rule files. To start a proxy, you set up rules for a proxy in your security policy and activate the policy.

Use of these proxies does not require installing additional client or server system software. However, some changes may be required in system configurations or user-supplied commands to access protected destinations through the proxies.

The activation process employs a script to see if the policy being activated contains one or more rules that use a given proxy. If so, the corresponding proxy is automatically started. If this same script determines that the Screen has been configured as a SecurID client, then the SecurID PIN server is started as well.

With SunScreen 3.2, filtering of scripts, applets, and viruses in downloaded content is possible using VirusWall, which is separately licensed from TrendMicro, Inc. See information about VirusWall scanning in “HTTP Proxy” on page 166, “SMTP Proxy” on page 173, and “VirusWall Content Scanning” on page 181.

The figure below shows a Screen using a proxy to filter packets for the HTTP protocol.

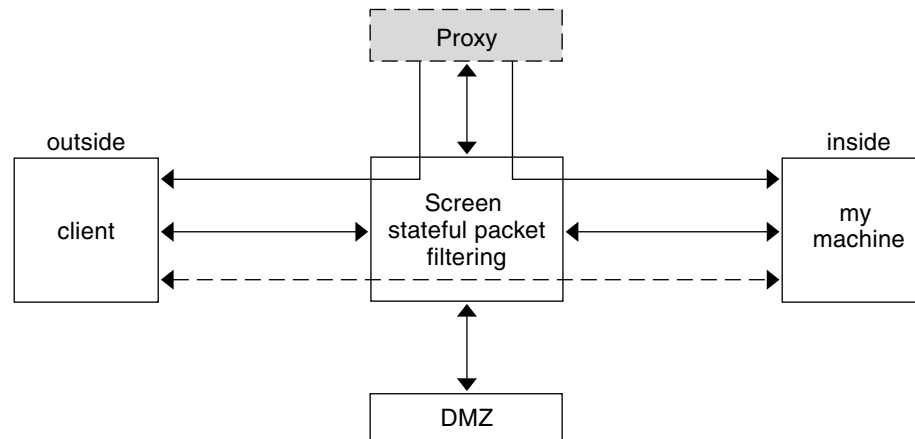


FIGURE 10–1 Screen With a Proxy

How Proxies Work

Each proxy is a multithreaded program provided with SunScreen. Each is configured through common objects and policy rules. A particular proxy is initiated or reconfigured whenever a policy is activated that contains rules that specify its type of access regulation.

Note – SunScreen proxies require the facilities of TCP and UDP internet protocols. As such, only Screens with at least one routing-mode (IP-address-bearing) interface can provide proxy services.

Policy Rule Matching

Each proxy is interposed in the middle of rules that reference it by the rule compilation process. In proxy rules, as in other Screen rules, you refer to originating client and destination (or *backend*) server address objects, not the Screen itself.

Note – The Screen rule compilation process actually produces two subrules for each proxy-use rule; one that allows client access to the proxy server program, and one that allows the proxy server program access to the backend servers. These rules are hidden. They effect a similar kernel-based stateful filtering mechanism to other rules for the Screen. The proxies themselves employ the original proxy-use rules for their own, more detailed, access control mechanisms.

In addition to stateful packet filtering within the Screen kernel, each proxy performs additional rule processing to control access. The additional checking enforces end-to-end (client-to-server) address and service matching, as well as user authentication, command restriction.

The general flow of tests that each proxy applies to gain access to requests is described below:

1. For FTP, Telnet, and (optionally) HTTP proxies: Has the user been properly authenticated?

Note – User authentication occurs only once, regardless of the number of rules configured. Authentication is based upon the user identity and accompanying passwords (if any) supplied by interaction with the client host. See Chapter 9.

2. For each rule configured for a particular proxy: Is the requested service port one that is handled by the proxy?

The proxy only receives connection requests for the ports on which it has been configured to listen.

1. Is the address of the client contained in the set of source addresses for the policy rule?
2. Is the address of the backend server (if applicable) in the set of destination addresses for the policy rule?

3. For HTTP URL references with specific port numbers: Is the target service port allowed by the proxy?
4. For FTP, Telnet, and (optionally) HTTP proxies: Is the authenticated user a member of the GROUP proxy user specified in the rule?

As with other Screen rules, these tests are performed in the order in which they appear within the policy. The first rule that matches all tested criteria takes effect with respect to any incoming request for a proxy-provided service. If no rule is found that matches all (applicable) criteria, the requested access is denied.

Proxy User Authentication

The FTP, Telnet, and (optionally) HTTP proxies of SunScreen provides the ability to restrict access to users who can verify their authenticity.

User authentication mechanisms of SunScreen are described in detail in Chapter 9. In this section, the discussion is prefaced by notes that pertain especially to how these user mechanisms are employed by the proxies.

The goals of user authentication within a proxy are to:

- Establish the authenticity of a proxy user
- Locate a rule that references that authenticated proxy user

A side-effect of establishing an authentic user is a collateral mapping to a *backend* user identity. This identity is a string that is supplied (by the FTP proxy) as the user of the backend server (for example, a user's userid on Solaris).

The second goal is achieved by the rule matching steps previously described. A rule that references the authentic proxy user itself, or that references a GROUP proxy user that contains an ENABLED member reference to that authentic proxy user, causes a successful user match.

Proxy Limitations

Proxy implementation has the following limitations:

- Proxies can only be used on Screens with at least one routing-mode interface.
- You cannot use proxies on Screens in an HA cluster.

Automatically-Saved Common Objects

The proxies use the following common objects:

- Authorized user
- Proxy user
- Jar hash
- Jar signature

Once these objects are added or edited, the change is stored immediately and cannot be reversed. The Save button in the administration GUI is greyed out to show that it is inactive. Although the changes made to these objects are saved immediately, they do not take effect until a policy is activated.

FTP Proxy

The FTP proxy functions as a relay for the File Transfer Protocol to enable you control connections based upon source and destination addresses and user authentication. It can also limit access to certain file transfer commands, such as `put` and `get`, based on source or destination addresses and user authentication.

You can configure the FTP proxy to ask for user authentication as an additional mechanism for controlling access to sites and commands.

FTP Proxy Operation

When the FTP proxy starts, it reads its policy files and then listens on the standard FTP port (port 21) for connections. When a connection is made, the FTP proxy starts a new thread to handle the connection, and the main thread returns to listening for other connections.

The child thread generates an FTP login banner and asks for a user name and password pair. The user name format is *proxyuser@server*. The password format is *proxypass@serverpass*, where *proxypass* is the password for the proxy, and *serverpass* is the password for the destination FTP server.

The FTP proxy validates the *proxyuser* name using *proxypass* as was described previously. The hostname (backend server), given in the `USER` command after the first `@` character, is translated to its IP address using the hostname-to-address translation mechanism configured for and in the context of the FTP proxy. The resulting addresses provide the values to use as matching criteria for the destination addresses in the proxy rules.

The standard proxy rule matching is used (see “Policy Rule Matching” on page 161). If a match is found, a connection is established to the FTP server of the user-requested destination. If multiple addresses result from the translation of the user-specified backend server, they are each tried in the order yielded by the name translation mechanism (for example, DNS).

Once a connection to the backend server is established, the proxy attempts to log in using the backend username generated during authentication and using `serverpass` as the password (see “Proxy User Authentication” on page 162). Once the backend user identity is established, commands that are allowed by flags associated with the policy rule in use are relayed, results returned, and files exchanged.

The following example illustrates a session between an FTP connection to a target host (`ftp.cdrom.com`) using anonymous FTP.

```
#ftp screen
Connected to screen
220- Proxy: SunScreen FTP Proxy Version 3.2
      :Username to be given as proxy-user@FTP-server
      :Password to be given as proxy-user@FTP-server-password@
220 Ready
Name (screen:edison): anonymous@ftp.cdrom.com
331-Proxy: Authenticate & connect
331 Password needed to authenticate 'anonymous'.
Password: [password is not echoed]
      :Authentication mapped 'anonymous' to backend user 'anonymous'.
      :Connecting to ftp.cdrom.com (165.113.121.81) - done
Server: 220 wcarchive.cdrom.com FTP server (Version 2.0) ready
Proxy: Login on server as 'anonymous'.
Server: 331 Password to server.
Proxy: Supplying password to server
230-Server:
230-Welcome to wcarchive - home ftp site for Walnut Creek CD-ROM
230-There are currently 2273 users out of 2750 possible
230 Guest login OK, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

FTP Proxy and Anonymous FTP

The proxy user `anonymous` is configured during the installation process as an unauthenticated proxy user. As such, any string provided before the first `@` in the password is ignored. The password after the first `@` in the password sequence (that is, `edison@carter.com`) is the backend user password, which, for anonymous FTP, is traditionally the user's email address.

FTP Proxy Use

To use the proxy and make FTP connections, the user must open an FTP connection to the Screen rather than open a direct connection to the end system. The Screen's policy rules only allow FTP connections to and from the FTP proxy.

Other FTP Proxy Issues

The FTP proxy does not permit the PASV command (used for third-party transfers).

The FTP proxy has a 10-minute time-out on the control connection for user requests. The responses from the backend server have only two minutes to arrive before timing out.

The maximum number of concurrent sessions available in the FTP proxy daemon is configurable through the variable `N_Sessions`. It contains the following items:

- `sys=Screen` (optional)
- `prg=ftpp`
- `name=N_Sessions`
- `values=max # of sessions`
- `description="descriptive text"` (optional)
- `enabled | disabled` (The default is enabled)

As initially installed, a global version of this variable is created that restricts the number of concurrent sessions to 100.

The following is an example of what you would type to display this (initial) variable while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=ftpp name=N_Sessions
PRG="ftpp" NAME="N_Sessions" ENABLED VALUE="100"
DESCRIPTION="limit # of concurrent sessions, FTP proxy"
```

You can alter this number of sessions, perhaps to be more restrictive, on a particular Screen.

The following is an example of what you would type to do this while logged into the primary Screen:

```
edit> vars add sys=Screen prg=http name=N_Sessions value=66
description="limit # of concurrent sessions on the Screen FTP proxy"
edit> quit
```

By configuring the FTP proxy on a Screen, the actual FTP service into that system becomes unavailable. To avoid confusion, define the destination address of proxy rules to exclude all the addresses of Screens. (You can still FTP out of the Screen, as necessary.)

HTTP Proxy

The HTTP proxy as currently implemented is a means of filtering Web content for outbound references from Web browsers to external, unknown servers.

The HTTP proxy:

- Provides a relay capability for HTTP access to the World Wide Web; this relay supports HTTP and the Secure Sockets Layer (SSL).
- Allows or denies sessions based on the source and destination addresses
- Provides selective filtering of HTTP content (such as Java applets, ActiveX, and cookies) based on the source and destination addresses for sessions
- Provides for optional user authentication
- Can be configured to perform content filtering
- Is useful in implementing network address translation (NAT) because it originates all outbound connections to Web servers; it reuses a single IP address for multiple browser clients.
- Can filter Java applets based on the signatures encapsulated in Jar files attached to the applet or based on a precomputed hash of valid applets
- Is configured on a per-rule basis in the configuration editor to:
 - Remove *cookies* (see “HTTP Proxy Limitations” on page 173)
 - Block ActiveX content
 - Block or allow all Java content (see “HTTP Proxy Limitations” on page 173)
 - Block SSL content.
- Can be used with virus-scanning software

Note – When you use your HTTP proxy in conjunction with VirusWall scanning, your HTTP proxy examines Web content for malicious scripting language code and executable content, and scans downloaded data for possible computer viruses. The scanner instructs that the content be blocked, or altered (for example, clean viruses from the content), or returned unaltered. Scanning results are recorded in the SunScreen log entries regarding HTTP transfers. See “VirusWall Content Scanning” on page 181 for detailed information about VirusWall.

To use the HTTP proxy, define the source addresses that should be allowed to have access to the proxy and to the destination addresses of allowed Web servers to be contacted. To create the HTTP proxy rule, use the built-in service `www`, reference your desired source and destination addresses, and select the `PROXY` for `PROXY_HTTP` during rule definition (in the administration GUI) or the `PROXY_HTTP` keyword (for command-line rule creation).

To require user authentication of HTTP traffic through the proxy, the `USER` keyword may be used in rule creation on the command line; equivalently, a `proxyuser` object may be associated with `PROXY_HTTP` rules within the rule manipulation facilities provided by the Admin GUI. This usage is similar to that of the FTP and Telnet proxies, but for the HTTP proxy, user authentication is optional.

HTTP Proxy Port Restrictions

The HTTP proxy allows further restrictions based on target port numbers. HTTP provides for user-specified references to arbitrary port numbers using the `http://host:port/...` construction. Because of the way in which the Screen operates, client browsers can be inadvertently allowed access to services that would otherwise be restricted save for this feature of HTTP.

The port restriction mechanism for the SunScreen HTTP proxy is controlled by the variable `TargetSvcs`. This variable can either be global or Screen-specific. It contains the following items:

- `sys=Screen` (optional)
- `prg=http`
- `name=TargetSvcs`
- `values={ svc=service... }` – Name or names of service objects for services to allow
- `description="descriptive text"` (optional)
- `enabled | disabled` – The default is `enabled`.

As initially installed, a global version of this variable is created that restricts such access to port 80 (the `www` service).

The following is an example of what you would type to display this (initial) variable while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=http name=TargetSvcs
PRG="http" NAME="TargetSvcs" ENABLED VALUES={ svc="www" }
DESCRIPTION="target TCP ports that the HTTP proxy can get to"
```

To have access to a wider range of ports configuring a new service object and an added variable (for example, this time, for a particular Screen) is one way of doing this. The following example show what you type while logged into the primary Screen:

```
>admin% ssadm -r primary edit Initial
edit> add service www-targets SINGLE FORWARD "tcp" PORT
1024-1520 PORT 1522-3850 PORT 3856-5999 PORT 6064-65545 COMMENT "more TCP
port numbers to allow as targets in HTTP proxy URIs"
edit> vars add sys=screen prg=http name=TargetSvcs values={ svc=ssl
```

```
svc=www svc=www-targets } description="target TCP ports that the HTTP proxy can get to"
edit> save
edit> quit
```

Note – The above definitions prevent access to ports that are below 1024 except for `www` and `ssl`. It also prevents access to port 1521 (*SQLNET*), ports 3851 through 3855 (the Screen's administrative and SecurID PIN server ports), and ports 6000 through 6063 (which are used by X Windows). Tailor your restrictions to suit your security needs.

You can configure the SunScreen HTTP proxy to restrict Web content. You can block or allow content, or you can be configured to verify certain content (Java applets) based on digital signatures or hashes. You configure these content-filtering features as part of the rules employing the HTTP proxy.

HTTP Proxy Access for `ftp://`

The HTTP proxy relays access for the `ftp://` method to the FTP proxy. This approach enables users of the browser to list directories and download files. Most often, this facility is used in conjunction with URLs embedded in web content that is designed to facilitate file downloading.

The standard form of an `ftp://` method URL is as follows:

`ftp://user:passwd@host:/dir...type`

The *user*, *passwd*, and *type* are optional (and not often used by `ftp://` method references.) The list of *dir* components specifies path name of the reference. Note that SunScreen does not implement the *port* option for `ftp://` URLs.

The default behavior of the *user* and *passwd* references is to use anonymous FTP. A typical URL would then look like:

`ftp://codebloat.com/pub/dwnlds/explorer5.exe`

Control over the defaulting of *user* and *passwd* is obtained using three variables — `FtpPwdDomain`, `FtpPwdUser`, and `FtpUser` — each of which is described below.

`FtpPwdDomain` contains the following items:

- `sys=screen` (optional)
- `prg=http`
- `name=FtpPwdDomain`
- `value=domain` – *domain* is used in creating an anonymous password.

- `description="descriptive text"` (optional)
- `enabled | disabled` – The default is `disabled`.

This variable (if not found or disabled) in the HTTP proxy defaults at run-time to the domain name of the Screen (for example, the output of the Solaris `defaultdomain` command).

`FtpPwdUser` contains:

- `sys=screen` (optional)
- `prg=http`
- `name=FtpPwdUser`
- `value=user` – `user` is used in creating an anonymous password. The default is `webproxy..`
- `description="descriptive text"` (optional)
- `enabled | disabled` – The default is `enabled`.

The HTTP proxy uses this variable, in conjunction with the (perhaps defaulted) value of `FtpPwdDomain` to construct an anonymous password of the form:

`@FtpPwdUser@FtpPwdDomain`

If the `passwd` supplied in the URL is a string that contains no `@` characters (encoded as `%40`), then that `passwd` string takes the place of the value of `FtpPwdUser` in the anonymous password construction. Thus the URL:

`ftp://:bob@codebloat.com/pub/dwnlds/exploder5.exe`

enables designating the user `bob` for the user portion of the anonymous password (for example, `@bob@FtpPwdDomain`).

The value of `FtpPwdUser` can be changed to redirect potential email responses from FTP servers to an appropriate storage receptacle.

`FtpUser` contains:

- `sys=screen` (optional)
- `prg=http`
- `name=FtpUser`
- `value=proxyuser` – `proxyuser` is used in anonymous access. The default is `anonymous`.
- `description="descriptive text"` (optional)
- `enabled | disabled` – The default is `enabled`.

This variable does not normally need to be altered unless the identity of the predefined user `anonymous` in the Screen's `proxyuser` database has been changed.

In addition to the anonymous usage, it is possible to download using authenticated users through the FTP proxy using the `ftp://` method. An example URL is:

```
ftp://proxyuser:proxypass%40serverpass@server/README.TXT
```

This performs an operation identical to a direct interaction with the FTP proxy, supplying the user as `proxyuser@server` and the password as `proxypass@serverpass`.

To enable the HTTP proxy to use the `ftp://` method, one or more rules are needed to allow the HTTP proxy itself to be a client of the FTP proxy. The HTTP proxy always connects to the FTP proxy using the LOOPBACK (127.0.0.1) address. So, for example, to enable the `ftp://` method anonymous access to "outside" web servers

```
edit> add address outside ...
edit> add address local127 RANGE 127.0.0.1 127.255.255.255 ...
edit> add rule ftp local127 outside ALLOW PROXY_FTP USER anonymous FTP_GET FTP_CHDIR
```

This enables anonymous FTP through the proxy for any user on the Screen itself as well.

HTTP Proxy User Authentication

As previously stated, the HTTP proxy can optionally require users to authenticate themselves. The addition of a `USER` attribute to a rule, along with an associated `proxyuser` object name, will cause requests to require authentication (based on the source and/or destination addresses within the rule).

HTTP proxy authentication uses the same mechanisms as other proxies. See Chapter 9 for information about SunScreen user authentication mechanisms.

The SunScreen HTTP proxy supports the `Basic` HTTP authentication method and provides several HTTP proxy authentication variables for configuring the method.

These variables do not normally need alteration, and come prefigured for immediate operation. The variables are:

- `sys=Screen` (optional)
- `prg=http`
- `name=AuthBasicRealm`
- `value=SunScreen` (optional - prefigured as shown)
- `description="descriptive text"` (optional)
- `enabled | disabled` (prefigured as enabled)

The `Basic` authentication mechanism allows use of a *Realm* name; this realm name is displayed by the user's browser as part of the authentication, and is intended to be useful in identifying the collection of resources being protected by authentication. It is an authentication mechanism between the user and the programs he or she wants to

access. The administrator knows about this and other kinds of authentication associated with a firewall, but an end user may not. The realm variable identifies who or what is interposing this authentication on the proxy. What the user sees in the password process of the browser varies depending on how these variables are set. The word “SunScreen” could be replaced with some company-specific term, for instance.

- `sys= Screen` (optional)
- `prg=http`
- `name=AuthBasicTTL`
- `value=authttl` (default is “15m”)
- `description=“descriptive text”` (optional)
- `enabled | disabled` (prefigured as enabled)

`authttl` is a timeout, constructed of an unsigned number, followed optionally by one of the scaling characters (d, h, m, or s). These indicate time units of days, hours, minutes, or seconds, respectively. The scaling characters are case-insensitive, and the default unit scale is seconds.

Credentials for proxy authentication are stored by the browser and offered repeatedly with subsequent requests. This functionality prevents annoying the user with *many* such challenges, which would otherwise render web browsing unusable. However, the SunScreen administrator is allowed to place a reasonable upper limit on the amount of time a given browser instance (or group of instances) will be allowed to reuse the same credentials without requalification. Tailoring this variable restricts the time which an unattended browser *might* be usable by another party before the latter would be required to re-authenticate. The `AuthBasicTTL` variable controls this time-to-live setting for credentials using the Basic authentication method.

HTTP Proxy Operation

When the HTTP proxy starts, it reads its policy files, starts a Java Virtual Machine (JVM) for processing Jar (Java archive) files, and then listens on the standard HTTP port (80) for connections. When a connection is made, the HTTP proxy starts a new thread to handle the connection, and the main thread resumes listening.

The child thread reads the first line of the HTTP header request from the client and parses the actual destination address. The method (for example, `http://` or `ftp://`) is determined. If the `ftp://` method is requested, the request for file retrieval is forwarded to the FTP proxy as described above. Otherwise (for the `http://` method) the thread searches the proxy policy rules for a match on the source and destination addresses.

- If the rule matched specifies (proxy) user authentication, and if proper user credentials are not present in the request or have expired, the proxy returns a response indicating a need for (proxy) user credentials.

- If a match is found, the flags associated with that policy rule are set. The proxy then reads the rest of the client's HTTP request. If a match for the connection is not found, the HTTP proxy sends the client an error (Method Not Implemented) and closes the connection.
- If cookies are not allowed, any cookie responses are deleted.
- If SSL is allowed and the connection uses SSL, the connection continues. No further processing can take place, because the rest of the connection is encrypted. If SSL is not allowed, the HTTP proxy drops the connection and sends an error (Method Not Implemented) to the client.
- If the rule matched specifies (proxy) user authentication, and if proper user credentials are not present in the request or have expired, the proxy returns a response indicating a need for (proxy) user credentials.
- If the connection is allowed, the HTTP proxy attempts to open a connection to the actual destination web server. If the web server cannot be reached, the proxy returns an error (Cannot locate host) to the client and closes the connection.
- If the HTTP proxy connects to the web server, it sends the client's HTTP request to the server and waits for a response. When a response arrives, the proxy reads the response header and, if appropriate, drops any cookie requests. If the response header indicates that the response includes Java or ActiveX content, the proxy examines the response body to determine the nature of the content. If the content is not allowed, the proxy sends an error (Cannot connect to host) to the client and drops the connection.
- If content scanning has been configured, and once the above-mentioned proxy-based content checks have been performed, the resulting content is passed to the scanner for inspection. The scanner may instruct that the content be blocked, or may alter the content (clean viruses, for example) , or may return it unaltered. Scanning results are given to the user (as being blocked, if so determined). Scanning results are reflected in SunScreen log entries regarding the HTTP request and its results.

Java Virtual Machine (JVM)

If the body contains a Jar file, the Jar is passed to the JVM, which extracts the signature components and calculates a hash for the Jar. The signatures are compared to the list of approved signatures and if the signatures match and signed Jars are allowed, the data are passed on to the client. If hashed Jars are allowed, the proxy compares the computed hash to the list of approved hash values and if a match is found, passes the response to the client.

After all of the data have been passed to the client, the proxy closes the connections to the client and the server and terminates the thread.

Jar Hashes and Signatures

The HTTP proxy is configured on a global basis in the configuration editor to verify the list of acceptable signatures and hash values on Jar content.

Both the Jar hashes and signatures are stored in the `vars` database.

Note – Currently, there is no provision to create Screen-specific versions of the Jar hashes and signatures for the Screen

Jar Hashes

The following is an example of what you type to manage Jar hashes:

```
edit> jar_hash parameters
```

You can add, delete, list, and rename Jar hashes. You assign names, which are ephemeral strings. Names are used only to reference items when managing them using the `jar_hash` command.

Jar Signatures

You can add, delete, list, or rename Jar signatures. You assign the names, which are ephemeral strings. The names are used only for purposes of reference to items when managing them using the `jar_sig` command.

HTTP Proxy Limitations

Jar validation facilities acquire the hash and signature values that are added to the configuration.

ActiveX filtering currently blocks certain types of Java content.

SMTP Proxy

The SMTP proxy provides a *basic* level of control over incoming electronic mail that is based on the Simple Mail Transport Protocol (SMTP). It can be configured to allow or deny such email based on source addresses, the actual server name of the source host,

or the server name of the (claimed) originator of a message. It can be further configured to allow or deny relaying of email based on the destined host or server of a message.

The SMTP proxy can be configured to scan the content of incoming electronic mail messages for virus and other malicious content.

Note – When you use your SMTP proxy in conjunction with VirusWall scanning, your SMTP proxy examines email attachments for malicious scripting language code and executable content, and scans downloaded data for possible computer viruses. The scanner instructs that the content be blocked, or altered (for example, clean viruses from the content), or returned unaltered. Scanning results are recorded in the SunScreen log entries regarding SMTP transfers. See “VirusWall Content Scanning” on page 181 for detailed information about VirusWall.

SMTP Proxy Operation

When the SMTP proxy starts, it reads its policy files, determines its local server name for use in relay checking, and listens on the standard SMTP port (25) for connections. When a connection is made, the SMTP proxy starts a new thread to handle the connection, and the main thread resumes listening.

The child thread takes control of the connection from the client. It then attempts to reverse-translate the address of the client (from the connection state) to yield a registered name.

If a registered name is discovered, the suffixes in the `mail_spam` list are checked against that name. If a suffix matches (the end of) the name of the originating host, the connection is closed with a response (455) refusing reception.

If no name is registered for the address, then the address itself is sought in the `mail_spam` list (looking for items that contain a single address or a range). If a match is found, the connection is closed with a response (455) refusing reception.

If it passes the peer-address check, the proxy thread next attempts the typical proxy rule match steps (“Policy Rule Matching” on page 161), except that only the source address is checked. For each rule that matches, an SMTP connection is attempted to the message transfer agents(MTA) listed as destination for the rule.

Once a connection to a destination server MTA has been established, data are relayed between the client and server MTAs. The content is scanned for commands that introduce source mailbox, destination mailboxes, and the data stream itself. Source mailboxes are checked against the spam list (if any). The destination mailboxes are checked against the relay list.

If configured for content scanning, the body of the e-mail messages which pass the above-mentioned spam and access control mechanisms, are fed to the scanner for inspection. The scanner may instruct that the content be blocked, or may alter the content (clean viruses from it, for example) , or may return it unaltered. Scanning which results in content alteration is reflected in the e-mail messages so modified.. Scanning results are recorded in the SunScreen log entries regarding SMTP transfers.

Spam Control

Unsolicited electronic mail is colloquially known as "spam." The restriction of mail based on originator is known as spam control. The SMTP proxy provides the ability to define a list of one or more restrictors that operate based on either server name (suffix) or nonserver (address range) criteria.

Spam restrictors have one of two syntactic forms:

- *server suffix* (suffix in a named host), or
- *start address* [. . *end address*] (range of one or more IP addresses of unnamed hosts)

Note – No spaces are permitted around the double dots (. .) in this construct for the address range.

server suffix is simply an ASCII character string.

See "SMTP Proxy Operation" on page 174 for details regarding how these restrictors are used.

Spam restrictors are defined using the configuration editor, and the `mail_spam` subcommand of `ssadm edit` in the administration GUI.

Examples

Below are examples of working with spam restrictors.

- To display the current set of spam restrictors while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> mail_spam list
"total-nonsense.org"
"0.0.0.0..255.255.255.255"
```

This example listing shows two entries:

- One to refuse email from the server or servers with registered address in the domain `total-nonsense.org`
- The other to refuse mail from any host that does not have a registered server name (in a reverse-mapping of IP address to DNS name).
- To add an additional restriction while logged into the primary Screen:

```
edit> mail_spam add complete-spam.net
edit> quit
```

- The following removes a restriction while logged into the primary Screen:

```
edit> mail_spam delete lite-spam.com
```

- The following is an example of what you type to define spam restrictors to deny mail from some mail originators you know to be sources of unsolicited mail:

```
edit> mail_spam add 0.0.0.0..255.255.255.255
edit> mail_spam add dialups.naughty-isp.net
```

Relay Control

In addition to controlling incoming spam destined for your site, another important area of control over email is the limitation on accepting email and then relaying it to another location. Relayed mail is responsible for a great deal of the unsolicited email on the Internet. Improper relaying makes spam harder to defeat and leaves the relaying site open to various types of reprisal from the ultimate recipient-victims.

The SMTP proxy allows the configuration of a set of strings that, coupled with policy rules, enables you to restrict the destination domains that the proxy accepts.

Relay restrictions have one of two syntactic forms:

- *domain suffix* (suffix in recipients to allow)

or

- *!domain suffix* (suffix in recipients to disallow)

domain suffix is simply an ASCII character string.

See “SMTP Proxy Operation” on page 174 for details regarding how these restrictors are used.

Relay restrictors are defined using the configuration editor, through the `mail_relay` subcommand of `ssadm edit`. The following is an example of what you type to display the current set of relay restrictors while logged into the primary Screen:

```
admin% ssadm -r master edit Registry
edit> mail_relay list
"your-domain.com"
```



```
"!private.your-domain.com"
```

This listing shows two entries, one to set a base domain to allow in recipients, the other to block a private subdomain in recipients.

The following is an example of what you type to add an additional restriction while logged into the primary Screen:

```
edit> mail_relay add !lists.your-domain.com
edit> quit
```

The following is an example of what you type to remove a restriction while logged into the primary Screen;

```
edit> mail_relay delete !test.your-domain.com
```

Note – If relay checking is enabled (for example, NO_RELAY) and yet no relay restrictors are configured, the SMTP proxy defaults to allow only the domain configured for the Screen itself as *the* valid domain for inbound mail.

The SMTP proxy rules should use the `smtp` service, and specify the `PROXY` to be `PROXY_SMTP` during rule definition in the administration GUI (or the `PROXY_SMTP` keyword for command-line rule creation). The `RELAY` (or `NO_RELAY`) flag is used to specify whether to perform unrestricted versus restricted relaying of mail. Use `NO_RELAY` in conjunction with the `mail_relay` restrictors shown above to effect relay control.

The following is an example of what you type, presuming that you already have the following objects defined:

```
admin% ssadm -r primary edit Initial
edit> list address
"mta-primary" HOST 1.2.3.4 ...
"mta-secondary" HOST 1.2.3.5 ...
"outside" GROUP { } { inside } ...
```

The following is an example of what you type to define an address group to contain all inside MTAs:

```
edit> add address mtas GROUP { mta-primary mta-secondary } { }
```

The following is an example of what you type to define relay restrictors to specify the servers (and perhaps hosts) to be allowed in recipient mailbox names:

```
edit> mail_relay add prime-server.com
edit> mail_relay add other-server.com
edit> mail_relay add !lists.prime-server.com
edit> mail_relay add !private.other-server.com
```

Finally, the following is an example of what you would type to define a rule to cause inbound email to pass through the SMTP proxy:

```
edit> add rule smtp outside mtas ALLOW PROXY_SMTP NO_RELAY
edit> save
edit> quit
```

Note – Because of the way that SunScreen represents addresses (in numerical order by IP address), rules that refer to multiple destination addresses attempt the MTA connection in numerical order. To obtain finer control over MTA connection ordering, use multiple rules.

Once a connection is established to a willing backend MTA, the proxy thread begins watching information passed by the client to the server (ordinarily, the proxy does not talk to the client). The proxy looks for a `MAIL FROM:` command from the client. (This command gives the name of the user who originated the message. It is often abused by spam message creators, so its information is often untrustworthy.) The mailbox name in this command is compared with the suffixes in the `mail_spam` list and if found there, the connection is aborted with a response (455) refusing reception.

Next, the proxy thread looks for one or more `RCPT TO:` commands from the client. (Such commands give one of the destination mailboxes to which the message is directed. Unlike the `MAIL FROM:` command, this mailbox name is *always* a real recipient.) If the rule specifies `NO_RELAY`, then the mailbox names in these commands are compared against the `mail_relay` list. The search can be conceptually thought of as a two-pass search. On the first pass, any denial suffixes (ones beginning with `!`) are sought and if matched, cause a connection abort with a response (454) refusing reception. On the second pass, allowance suffixes are sought (ones *not* beginning with `!`). If one matches, the recipient is allowed; if none matches, the connection is aborted with a response (454) refusing reception.

SMTP allows multiple messages (each with one or more recipients) to pass on a single connection. Barring a refusal of service, once all messages have been passed, the proxy closes the connection to the client and backend server and ends the child thread.

Other Mail Configuration Issues

In addition to specifying SunScreen SMTP proxy policy, configuration steps must be taken to cause SMTP-based email to arrive at the proxy for delivery to the servers it is to represent. The typical mechanism is to create MX records in the Domain Name System (DNS) for those servers. The procedure for altering DNS configuration is outside the scope of this document, but should be very familiar to your DNS administrator.

More than one Screen can be configured to operate in parallel to service incoming mail. The centralized management group feature of SunScreen makes doing so a relatively straightforward task. These parallel Screens can then be used within MX records.

By configuring the SMTP proxy on a Screen, the actual SMTP (inbound email) service into that system becomes unavailable. (You can still send email out of the Screen, as necessary.)

SMTP Proxy Rules

After configuring the content filtering, the final step is to create one or more rules that allow SMTP-based email to be served. A typical configuration often allows filtering outside email through the proxy to one or more inside message transfer agents (MTA) such as Solaris mail servers.

The source address for SMTP proxy rules should allow mail from any potential mail-sending address. Restrictions on source addresses in the rule can be useful in blocking abusive mail-originating sites that are stationary with respect to IP addressing.

The destination address for SMTP proxy rules should contain the addresses of one or more MTAs to which the proxy will connect to store and forward the incoming email.

Telnet Proxy

The Telnet proxy provides a virtual terminal relay and allows or denies connections based on source and destination addresses. The Telnet proxy can ask for user authentication as an additional mechanism for controlling access to sites and commands. The Telnet proxy does not support content filtering.

To authenticate users, the Telnet proxy can request a user name from a user and, based upon the type of authentication for that user name, request and validate a reusable password or digital token.

Telnet Proxy Operation

When the Telnet proxy starts, it reads its policy files and listens on the standard Telnet port (23) for connections. When a connection is made, the Telnet proxy starts a new thread to handle the connection, and the main thread returns to listening.

The child thread generates a proxy login banner and waits to read the user name and password. The format for user names consists of a login ID and a destination host separated by an @ symbol; for example, `lionel@manduck.bafb.af.mil`. The Telnet proxy validates the user name and password. If an invalid user name or password is sent, the Telnet proxy sends an error to the user and closes the connection. If the user name or password is valid, the source and destination addresses are checked against the Screen's policy rules. If a match is found, the flags associated with that policy rule are checked. If the connection is permitted, the Telnet proxy opens a connection to the actual destination server and relays data between the source host and the destination host.

The hostname (backend server) given in the user prompt, after the @ character, is translated to its IP address using the hostname-to-address translation mechanism configured for and in the context of the Telnet proxy. The resulting addresses provide the values to use as matching criteria for the destination addresses in the proxy rules.

The standard proxy rule matching is employed (see "Policy Rule Matching" on page 161). If a match is found, a connection is established to the Telnet server of the user-requested destination. If multiple addresses result from the translation of the user-specified backend server, they are each tried in the order yielded by the name translation mechanism (for example, DNS). Once a connection to the backend server is established, all data are relayed (uninspected) by the thread in both directions until either end terminates.

Other Telnet Proxy Issues

The Telnet proxy uses TCP *keep-alives* on the connection to the client. The Telnet proxy insists on doing its own echoing during the initial user authentication portion of the session. Some Telnet client programs do not obey standard Telnet echo negotiation and improperly double-echo `username@hostname` input and single-echo the password.

Note – By configuring the Telnet proxy on a Screen, the Telnet service into that system becomes unavailable. To avoid confusion, define the destination address of proxy rules so as to exclude all the addresses of your Screens. (You can still use the `rlogin` protocol to access the Screen, as necessary, and `telnet out`.)

Telnet Proxy Use

Before a client can connect to a remote host when the Telnet proxy is active, the client must first connect to the Telnet proxy. In the following example, the Telnet proxy is running on the host Screen, and the user wants to connect to the remote system `foo.com`.

```
riyal% telnet Screen
SunScreen Telnet Proxy Version: 3.2
Username@Hostname: edison@foo.com
```

At the password prompt, you type the password for the proxy authentication. The Telnet proxy would compare the specified user name and password to the list of valid proxy users and their passwords. If the user name/password are correct and the connection is allowed, the user is presented with a login banner for the machine `foo.com`.

Using Encryption With Proxies

You can have the Screen decrypt incoming traffic from a client before passing it to the proxy by creating two rules. The following is an example of using the Telnet proxy with SunScreen SKIP:

```
edit> add rule telnet proxycient localhost SKIP_VERSION_2 ...
edit> add rule telnet proxycient proxyserver PROXY_TELNET
```

Likewise, you can have the Screen encrypt the connection from the proxy to the backend server using a similar pair of rules:

```
edit> add rule telnet localhost proxyserver SKIP_VERSION_2 ...
edit> add rule telnet proxycient proxyserver PROXY_TELNET
```

VirusWall Content Scanning

This section describes how to configure your SunScreen HTTP or SMTP proxy to use the separately-licensed TrendMicro VirusWall content scanning option. Once you have installed and configured the VirusWall product on a server platform, you can direct your SunScreen HTTP or SunScreen SMTP proxy to use it for content examination. See “VirusWall Setup Issues” on page 186 for information about installing, configuring, and using VirusWall.

HTTP Proxy Access to VirusWall

Access to the VirusWall server from within the HTTP proxy is controlled by a service common object (`viruswall-http`) and a pair of variables (`VirusWallServerHTTP` and `scan.0`). The service object and variables are preconfigured as much as possible during installation, but the variables must be altered to activate the interface for scanning. In addition, you may need one or more access rules to allow your Screen

access to the VirusWall scanner server; see “To Add an Administrative Access Rule for Remote Administration” in *SunScreen 3.2 Administration Guide* for more information.

The `viruswall-http` service common object is employed by the interface between SunScreen’s HTTP proxy and VirusWall. The TCP service port defined for this object is set to that used by the default VirusWall installation.

VirusWallServerHTTP Variable

The `VirusWallServerHTTP` variable configures the interface between your Screen and VirusWall:

- (Optional) `sys=Screen`
- `prg=scan`
- `name=VirusWallServerHTTP`
- `values={ vwvalues }`
- (Optional) `description="descriptive text"`
- `enabled | disabled` (the initial configuration is enabled)

Options for the `vwvalues` portion of the `VirusWallServerHTTP` variable are:

- `type=VirusWall`
- `svc=viruswall-http`
- `addr=vwserver` (the default is the undefined address object, `viruswall-server`)
- (Optional) `holddown=downsecs` (the default is 30*60 seconds)
- (Optional) `maxconns=#maxconns` (maximum concurrent connections; the default is 3)
- (Optional) `minconns=#minconns` (minimum spare connections; the default is 1)

Note – Multiple `addr` items can be configured to allow the use of secondary scanning servers. Each `addr` item can designate address common objects by name or it can designate a naked (dotted-quad) IP address.

scan.0 Variable

The second variable, `scan.0`, connects the first variable to the proxy and specifies it as the first scanning facility to be used by that proxy. Because VirusWall scanning is optional, `scan.0` is predefined as `DISABLED`. To turn on scanning, you must set the `scan.0` variable to `ENABLED`.

- (Optional) `sys=Screen`
- `prg=http`
- `name=scan.0`
- `values={ scvalues }`

- (Optional) `description="descriptive text"`
- `enabled | disabled` (the initial configuration is disabled)

For the variable `scan.0`, the *scvalues* portion is `name=VirusWallServerHTTP`. Note that the value of this variable is the name of the first variable.

Both variables—`VirusWallServerHTTP` and `scan.0`—are pre-defined. If you display their values from the command line configuration editor, you see:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=scan name=VirusWallServerHTTP
PRG="scan" NAME="VirusWallServerHTTP" ENABLED
VALUES={ type="VirusWall" svc="viruswall-http" addr="viruswall-server" }
DESCRIPTION="TrendMicro HTTP scanning server(s)"
edit> vars print prg=http name=scan.0
PRG="http" NAME="scan.0" DISABLED VALUES={ name="VirusWallServerHTTP" }
DESCRIPTION="HTTP proxy content scanner"
```

HTTP Access Rules

One or more access rules may be needed to allow your Screen access to the VirusWall scanner server (see “To Add a New Rule” in *SunScreen 3.2 Administration Guide*).

Because VirusWall scanning is optional, and because the `viruswall-server` address object cannot be preconfigured during installation, the following example shows prototypical post-installation steps to enable VirusWall scanning of HTTP content:

```
admin% ssadm -r primary edit Initial
edit> add address viruswall-server 10.73.176.13
edit> add rule viruswall-http localhost viruswall-server ALLOW
edit> add rule www 'inside' web-scanner ALLOW PROXY_HTTP
edit> vars add prg=http name=scan.0 ENABLED
VALUES={ name=VirusWallServerHTTP } DESCRIPTION="HTTP proxy content scanner"
```

This example:

- Defines the address for `viruswall-server`
- Adds a rule to allow communication between the Screen and the VirusWall scanner
- Adds another rule to allow HTTP proxy traffic
- Sets the `ENABLED` flag to turn on HTTP proxy content scanning

If content scanning has been configured, and once proxy-based content checks have been performed, the resulting content is passed to the scanner for inspection. The scanner may instruct that the content be blocked, or may alter (for example, clean viruses from) the content, or may return it unaltered. You receive scanning results (as being blocked, if so determined) that are reflected in SunScreen log entries regarding the HTTP request and its results.

SMTP Proxy Access to VirusWall

Access to the VirusWall server from within the SMTP proxy is controlled by a service common object (`viruswall-smtp`) and a pair of variables (`VirusWallServerSMTP` and `scan.0`). The service object and variables are preconfigured as much as possible during installation, but the variables must be altered to activate the interface for scanning. In addition, you may need one or more rules to allow SunScreen firewall access to a VirusWall scanner that is operating on a separate server platform.

The `viruswall-smtp` service common object is employed by the interface between SunScreen's SMTP proxy and VirusWall. The TCP service port defined for this object is set to that used by the default VirusWall installation.

VirusWallServerSMTP Variable

The `VirusWallServerSMTP` variable configures the interface between your Screen and VirusWall:

- (Optional) `sys=Screen`
- `prg=scan`
- `name=VirusWallServerSMTP`
- `values={ vwvalues }`
- (Optional) `description="descriptive text"`
- `enabled | disabled` (the initial configuration is enabled)

Options for the `vwvalues` portion of the `VirusWallServerSMTP` variable are:

- `type=VirusWall`
- `svc=viruswall-smtp`
- `addr=vwserver` (the default is the undefined address object, `viruswall-server`)
- (Optional) `holddown=downsecs` (the default is 30*60 seconds)
- (Optional) `maxconns=#maxconns` (maximum concurrent connections; the default is 3)
- (Optional) `minconns=#minconns` (minimum spare connections; the default is 1)

Note that multiple `addr` items can be configured, allowing the use of secondary scanning servers. Each `addr` item can designate address common objects by name, or may give a naked (dotted-quad) IP address.

scan.0 Variable

The second variable, `scan.0`, connects the first variable to the proxy and specifies it as the first scanning facility to be used by that proxy:

- (Optional) `sys=Screen`

- `prg=smtpp`
- `name=scan.0`
- `values={ scvalues }`
- (Optional) `description="descriptive text"`
- `enabled | disabled` (the initial configuration is disabled)

For the variable `scan.0`, the *scvalues* portion is `name=VirusWallServerSMTP`. Note that the value of this variable is the name of the first variable.

Both variables—`VirusWallServerSMTP` and `scan.0`—are pre-defined. If you display their values from the command line configuration editor, you would see:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=scan name=VirusWallServerSMTP
PRG="scan" NAME="VirusWallServerSMTP" ENABLED
VALUES={ type="VirusWall" svc="viruswall-smtp" addr="viruswall-server" }
DESCRIPTION="TrendMicro SMTP scanning server(s)"
edit> vars print prg=smtpp name=scan.0
PRG="smtpp" NAME="scan.0" DISABLED VALUES={ name="VirusWallServerSMTP" }
DESCRIPTION="SMTP proxy content scanner"
```

SMTP Access Rules

One or more access rules may be needed to allow your Screen access to the VirusWall scanner server.

Because VirusWall scanning is optional, and because the `viruswall-server` address object cannot be preconfigured during installation, the following example shows prototypical post-installation steps to enable VirusWall scanning of SMTP content:

```
admin% ssadm -r primary edit Initial
edit> add address viruswall-server 10.73.176.13
edit> add rule viruswall-smtp localhost viruswall-server ALLOW
edit> add rule smtp 'inside' mail-server ALLOW PROXY_SMTP
edit> vars add prg=smtpp name=scan.0 ENABLED
VALUES={ name=VirusWallServerSMTP }
DESCRIPTION="SMTP proxy content scanner"
```

If content scanning has been configured, and once the aforementioned proxy-based content checks have been performed, the resulting content is passed to the scanner for inspection. The scanner may instruct that the content be blocked, or may alter (for example, clean viruses from) the content, or may return it unaltered. You receive scanning results (as being blocked, if so determined) that are reflected in SunScreen log entries regarding the SMTP request and its results.

VirusWall Setup Issues

This section discusses the general issues of using SunScreen in conjunction with the VirusWall content scanning option once you have set up and configured your SunScreen HTTP or SMTP proxy.

Currently, SunScreen interoperates with VirusWall, version 3.32, for the Lucent Managed Firewall specifically. Only this version contains the necessary interface protocol that allows SunScreen to use the scanning facilities of VirusWall for HTTP or SMTP content. Aside from representing some hardware and software duplication issues, it also creates some additional security risks that you must minimize.

Windows environments are apt to imbed the need to run the Internet Explorer (IE) Web browser and can further require you to run Active-X as well as enable other executable content within the browser. Because Active-X and its kindred effectively run as root on an Administration Station, the potential for security compromise is immediately obvious. To minimize the potential for viral infection of the VirusWall platform, restrict the access that platform has to net traffic to the extent possible.

This restriction takes two forms:

- Physical network isolation
- VirusWall Internet access restriction

Place VirusWall on its own, separate SunScreen interface to effect physical isolation of the VirusWall platform. Should your system be compromised, this isolation defeats the possibilities that VirusWall:

- Denies the system a position of unfettered access to other systems
- Limits the potential to exploit various security holes found in NT server installations
- Could view traffic that might otherwise flow past it

To effect access restrictions, your system only needs to interact with other hosts in the following ways:

- Inbound access from the SunScreen firewall using it for content filtering
- Outbound name service (for example, DNS) access to resolve hostnames
- Outbound access to the TrendMicro server(s) from which it periodically downloads pattern files
- (Optional) Inbound access from your browser clients to the VirusWall scanner's Web server to allow you to retrieve (infected) content being held by the scanner
- (Optional) Outbound access to your SMTP server(s) through which the TrendMicro server(s) sends notification messages
- (Optional) Outbound access to other TrendMicro servers to browse TrendMicro's online documentation

- (Optional) Inbound access from a browser for remote administration of the VirusWall server

Only the first three access paths are mandatory for the scanning operation of the product, and only the first five access paths are mandatory for full operation of the product.

Note – It is recommended that you not use this system for any other purpose.

For you to effect the above security environment, contact TrendMicro for a definitive list of servers to which your VirusWall server needs access. Also, you can request written disclosures or privacy policies regarding all interactions between the VirusWall server you are deploying and TrendMicro's servers.

Once the Viruswall and related software is fully loaded, consult your product documentation or TrendMicro technical support for any questions regarding VirusWall configuration settings or options.

To test the access paths from the HTTP or SMTP proxy, browse the Web or cause inbound email to flow through your VirusWall-enabled SunScreen proxy. The SunScreen logs contain annotation of the added scanning activities.

Also, set LOG_SESSION on the rules to enable the downloading of pattern files from TrendMicro, and any other outbound connections you elect to allow for optional paths. More detailed information about pattern downloads can be obtained from the VirusWall configuration facilities (either Windows application or browser based).

Logging

Examination of logged packets is useful when you are trying to identify the causes of problems during network configuration or administration. You can also examine logs periodically for evidence of attempts to break into your network. This chapter contains information on:

- “Packet Logging” on page 189
- “Log File Locations” on page 190
- “Configuring Traffic Log Size” on page 190
- “Log Retrieval and Clearing” on page 196
- “Log Statistics” on page 198
- “Log Inspection and Browsing” on page 199
- “SunScreen we1fmt Utility” on page 201
- “HTTP Proxy Header Logging” on page 202
- “Logged Network Packet Enhancements” on page 203
- “General Event Type Enhancements” on page 204
- “Log Filtering Macros” on page 206

Packet Logging

SunScreen provides flexible logging of packets. This means that each Screen can keep a log of its traffic as configured. In HA clusters, a log of the packets is kept on the Screen that passed or rejected the packets.

You can configure a Screen to log a packet when the packet matches a rule or when it does not. Most frequently, packets matching DENY rules or packets that are dropped because they do not match any rule are logged. The action defined in a rule controls whether a packet is logged and what information about the packet is recorded.

Each system in a high availability (HA) cluster logs what that Screen passed or rejected, as well as local Screen events. Only the active Screen in an HA cluster logs packets, but even when an HA Screen is passive, some local events (such as becoming the active Screen) are logged.

Logging Limitations

The following limitations apply to logging:

- During a situation or time when there is excessive traffic through the Screen, not all packets are logged.

This logging limitation is an isolated instance and depends on how fast your system runs.

- Decrypted packets are logged, but SKIP certificate IDs are not logged.
- Only the active system logs packets.

If the active HA cluster Screen fails, its logs become inaccessible, and the new active HA cluster Screen begins logging the packets.

Log File Locations

SunScreen stores its log files in the following locations:

- The SunScreen packet, session, and extended event log is stored on the Screen in the `/var/SunScreen/logs` directory.
- SKIP logs are kept in the `/var/log/` directory.
- High availability (HA) event logs are logged according to the configuration settings for `syslog`.

Configuring Traffic Log Size

You can configure the size of the area used to log packet traffic, session and other events. The log consists of a number of files in a particular directory. Each Screen has a log file of its own.

You can specifically configure the size of the log file on each Screen. You establish the size of the log files in much the same way as other configuration items. These sizes are propagated to various Screens being managed during the normal activation process. The log file is resized on a particular Screen only when that Screen is restarted after the activation. This is true for primary and secondary Screens in a centralized management group and in an HA cluster.

You should change the size of the log file when you are configuring your policies just after installing the Screen. Activating the policy with the new log sizes does not resize the log files. The log file on a particular Screen is only resized when that Screen is restarted. When the Screen is restarted, it uses the policy that was the last currently active one. Instructions for setting logsize (global and for a particular Screen) are included below.

Setting the size of the log file does not cause the file system to allocate space for storing the log immediately. Competing users of the file system on which the log file resides should, therefore, not be allowed to consume this space. Even when the log has filled and begins to reuse filesystem space, the maximum amount of filesystem space is still not in use at all times.

Configuring the Global Default Log Size

The global default log size, which can only be configured using the configuration editor, is controlled by the variable `LogSize`. It contains the following items:

- `prg=log`
- `name=LogSize`
- `value=size` (in megabytes)
- `description="descriptive text"` (optional)
- `enabled | disabled` (default is enabled)

Examples: Setting or Displaying the Global Default Log File Size

Group Screen installations are configured on the primary Screen. To set the global default log file size to 250 Mbytes, while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars add prg=log name=LogSize value=250 description="log size (MB) "
```

To display the global default log file size:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=log name=LogSize
PRG="log" NAME="LogSize" ENABLED VALUE="250" DESCRIPTION="log size (MB) "
```

Note – Although the output produced by `print` surrounds the value of each item in double quotes, these are only necessary on *input* if there are embedded spaces within the values of items. Also, although `print` outputs all tag names in capital letters (for example, `PRG`), these tags are recognized in a case-insensitive manner on input (for example, `prg`, `Prg`, and `PRG` are equivalent).

Configuring the Log Size for a Specific Screen

This section shows you how to set a log size for a particular Screen that is different from the global log size.

Examples: Displaying and Setting the Log File Size

To display the log file size for all Screens in an HA cluster:

```
admin% ssadm -r primary edit Initial
edit> list Screen
scrn1 CDP ROUTING DNS
scrn2 CDP ROUTING DNS LOGSIZE 444
```

`scrn1` does not have the log file size configured and uses the global default value. `scrn2` has a size of 444 (Mbytes) that it uses instead of the global default value on that Screen.

To set the log file size for a specific Screen (`scrn1`) to 200 MB:

```
admin% ssadm -r primary edit Initial
edit> add Screen scrn1 CDP ROUTING DNS LOGSIZE 200
edit> save
edit> quit
```

Note – When altering the value of `LogSize`, be sure to reenter all the other attributes as they were displayed by the `list` verb.

Configuring Events to be Logged

Logs contain three basic types of events:

- Network traffic (packet)
- Network session summaries
- Extended events

Network Traffic (Packet)

You can set the action for each rule to be `ALLOW`, `DENY`, `ENCRYPT`, or `VPN`. For each action, you can set the kind of packet logging that you want:

- `LOG_NONE` – Do not log packets
- `LOG_SUMMARY` – Records the first 40 bytes of the packet in the log
- `LOG_DETAIL` – Records the complete packet in the log

Network Session Summaries

You can set the action to `LOG_SESSION` in a rule so that it records information about the session in the log. The information saved consists of the source and destination addresses and ports (if applicable), the amount of data being sent in each direction, and the length of the session. This action is not used for stateless services such as `ip all`.

The `SESSION` setting does not log packet content. Each basic protocol (for example, `IP`, `UDP`, `TCP`) logs statistics related to sessions as they are finalized.

This option is not available for the `DENY` action (because no session was allowed).

Extended Events

Other events are logged besides packets and sessions. They are stored in an extended format. These other events arise from the following logging entities:

- `auth` – Authentication logic (in various other agents)
- `edit` – Configuration editor
- `ftpp` – The FTP proxy
- `ha` – The high-availability subsystem
- `http` – The HTTP proxy
- `iked` – The IKE daemon
- `log` – The logger itself
- `smtpp` – The SMTP proxy
- `telnetp` – The Telnet proxy

Each entity has a `LogSeverity` variable which limits the extent of its logging of noteworthy events based on the severity level of those events.

In addition, there exist default limiters as catchalls for unnamed entities:

- `name=LogSeverity` – For all Screens
- `sys=Screen name=LogSeverity` – Screen-specific

The LogSeverity variables take text strings as their value. The value functions as a not-more-detail-than limiter and is similar to the functionality of the Solaris syslog command. The text values are:

- NONE
- ALERT
- CRIT
- ERR
- WARN
- NOTE
- INFO
- DEBUG

These limiter variables operate with several levels of globality, within entities and/or Screens, and/or universally. The limiters serve to control logging situations where a particular rule is not yet known to the entity, or where no particular rule applies.

In addition, the effect of the per-rule DETAIL, SUMMARY, and SESSION attributes is overridden by some of these logging entities. This override allows for finer control over events that can be attributed to a particular rule. Specifically, any rule-specific event of a severity of INFO or greater is logged if that rule has packet or session logging enabled.

Size of Logged Items

All items in the logs have a common, 24-byte header. After this header, the sizes shown in the table below apply to logged items (by type).

TABLE 11–1 Sizes of Logged Items

Type		Total Item Size (in bytes)
(packet)	DETAIL	24 + 44 + <i>size of packet</i>
	SUMMARY	24 + 44 + 40
SESSION	ip	24 + 40
	tcp	24 + 44
	udp	24 + 40
EXTENDED		24 + 64 + <i>UTF-8 text: 0 to 4008</i>

Level of Logging

Because the level of logging from a given program entity can be limited by the setting of the `LogSeverity` variable for that entity, a variable can be specific to a particular Screen (within a CMG group of centrally-managed Screens) or applicable to all Screens (those without a Screen-specific variable). If no variable is defined for a given entity, more general variables control such logging (again, on a per-Screen or non-Screen-specific basis). The search order for log limiter variables can be summarized as:

```
sys=Screenname prg=entityname name=LogSeverity
prg=entityname name=LogSeverity
sys=Screenname name=LogSeverity
name=LogSeverity
```

As initially configured, SunScreen contains log limiter variables for each program entity, as well as the non-Screen, non-entity-specific (*global global*) default. All are initially configured to the value `info`.

Log limiter variables are configured using the configuration editor.

Configuring Log Event Limiters

The log limiters are controlled by `LogSeverity` variables, which contain the following items:

- `sys=Screenname` (optional)
- `prg=programname` (optional)
- `name=LogSeverity`
- `value=severityname` (emerg,alert,...,debug)
- `description=descriptive text` (optional)
- `enabled | disabled` (The default is enabled.)

Once log limiters have been altered, the configuration must be activated to propagate the changes.

Examples: Manipulating Log Limiters

Do the following while logged into the primary Screen.

To Display the *global global* log limiter:

```
admin% ssadm -r primary edit Initial
edit> vars print name=LogSeverity
NAME="LogSeverity" ENABLED VALUE="INFO" DESCRIPTION="global log severity limit"
```

To display the *global* log limiter for authentication events:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=auth name=LogSeverity
PRG="auth" NAME="LogSeverity" ENABLED VALUE="INFO" DESCRIPTION="global log severity limit,
authentication"...
```

To log more debugging information on a particular Screen for authentication events:

```
admin% ssadm -r primary edit Initial
edit> vars add sys=Screenname prg=auth name=LogSeverity value=debug
description="debug authentication operations"

edit> quit
```

Note – Although, the output produced by `print` surrounds the value of each item in double quotes, these are only necessary on *input* if there are embedded spaces within the values of items. Also, although `print` outputs all tag names in capital letters (for example, `PRG=`), these tags are recognized in a case-insensitive manner on input (for example, `prg=`, `Prg=`, `PRG=` are equivalent). Finally, the `VALUE` string for the `LogSeverity` variable is likewise processed in a case-insensitive manner.

Log Retrieval and Clearing

A Screen's log is retrieved and can be cleared using the `log` subcommand of `ssadm`. Filtering can be applied by the Screen during retrieval, reducing the content of the log retrieved (see "Log Inspection and Browsing" on page 199).

Note – Clearing a Screen's log does *NOT* free up any disk space. The disk space used by the log, up to the maximum log size, is maintained, to guarantee that there will always be room for the log data. Note also that the disk space used by a log may be slightly more (approximately sixteen megabytes) than the configured log size.

You can retrieve and postprocess logs on an automated basis. (See the man page for `sslogmgmt` (1M) for details on automated log management.) `sslogmgmt` (1M) automates management of SunScreen logs for a group of centrally managed Screens. It can be run manually, as it is designed to run as a `cron` (1M) job. It is a feature installed by the `SUNWsfwau` package.

The table below shows the three subcommands for `log`.

TABLE 11–2 log Subcommands

Subcommand	Description
get	Retrieves the current content of the log, but leaves it intact.
clear	Makes the current content of the log inaccessible for further retrieval operations.
get_and_clear	Combines these two functions, clearing the log through the point to which it was retrieved.

Retrieval (get) of the log produces the (binary) log records on the standard output of `ssadm`. This can be redirected to a file or perhaps piped into other processes, such as `ssadm logdump` for viewing, further filtering, and so forth. The output stream from a `log get*` operation is manipulated with the piping and redirection capabilities of the shell that you are using on your Administration Station.

Note – The `get_and_clear` operation clears the log when the Screen producing the log finishes writing to the connection used to retrieve it. If the administration platform crashes, the disk to which the log is being written fills, or other exceptional conditions occur near the end of the retrieval, the clear phase can complete and some log records can be lost.

Examples: get, clear, get_and_clear

The following are examples of working with the `get`, `clear`, and `get_and_clear` commands.

To get the log from a Screen you are logged into (primary or secondary) and store the filtered result in a local file:

```
admin% ssadm -r Screen log get [filterargs] >local_log_file
```

See “Log Filters and the `logdump` Command” on page 199 for information about filtering.

To simply clear the log:

```
admin% ssadm -r Screen log clear
```

To get the log, store the filtered result in a local file, and clear the log:

```
admin% ssadm -r Screen log get_and_clear [filterargs] >local_log_file
```

The `get_and_clear` and `clear` verbs take an optional argument, `-U`: a designation of the user who cleared the log. This text string is displayed when log statistics are retrieved.

Here are examples of the `get_and_clear` and `clear` commands. To get and clear the log, setting the user designator as "ben":

```
admin% ssadm -r Screen log -U "ben" get_and_clear [filterargs] >local_file
```

To clear the log, setting the user designator as "Ben" and commenting that Ben should reset a counter or something after moving a file:

```
admin% ssadm -r Screen log -U "Ben: reset after move" clear
```

Log Statistics

You can inspect the current statistics of a Screen's log file from the administration GUI or by using the `logstats` subcommand of `ssadm`, as shown below.

Note – You must enter the name of the Admin Interface of the Screen as listed in the Naming Service or in the hosts file.

ssadm logstats Subcommand

To display the statistics for the log on a given (primary or secondary) Screen, while logged into that Screen:

```
admin% ssadm -r Screen logstats
Log size (bytes): 170600
Log maximum size (bytes): 60000000
Log loss count (records): 0
Cleared at: 04/01/1999 17:43 PST
```

Log size details the number of bytes currently contained in the log. Log maximum size gives the (rough) upper bound on the size of the log file. (It is the configured log file size multiplied by one million.) Both of these sizes indicate only the space to store the logged records, not the additional, minimal space needed to manage those records.

Log loss count gives the number of records that the logging server has discarded due to the log wrapping around before being retrieved or cleared.

Cleared at gives the date and time the log was last cleared. If a text designator (through the `-U` option to the `log` command) was given during the operation that cleared the log, then this line has the form:

```
Cleared by: ben at 04/01/1999 17:43 PST
```

Log Inspection and Browsing

Log mechanisms, processed through user-defined or ad hoc filters, provide the ability to inspect previously retrieved stored logs. The results are either stored as logs or converted to displayable text. Using the administration GUI, a Screen's active log file can be browsed in either a historical or live mode. You must enter the name of the Admin Interface of the Screen as listed in the Naming Service or in the hosts file.

Log Filters and the logdump Command

Screen log filtering employs a common mechanism and language, regardless of the context in which it is used. This mechanism and language is embodied in the logdump command. The logdump command is based on, and is a superset of, the snoop program, which is provided with the standard Solaris operating environment.

logdump can be used on an Administration Station to filter and inspect logs during active retrieval or on logs previously retrieved and stored. In conjunction with the logmacro facility, predefined filters can be employed to simplify and regularize routine log processing tasks.

The general usage for logdump is as a subcommand of ssadm. ssadm provides character-set translation between strings embedded in log events and the local character set of the Solaris system on which it runs.

Note – Although logdump is used directly as an ssadm subcommand, all other places in SunScreen where log filtering is allowed employ the same filter specification language. The examples in this section are prototypical of these other usage contexts.

Nominally, logdump input is either a log record stream directly from a possibly remote Screen, or captured log records from a file. This source of input is specified by the -i option.

Examples: logdump Command

To process (piped-in) records from the standard input:

```
admin% ssadm -r Screen log get | ssadm logdump -i- [output args] [filter args]
```

To process local file log record input:

```
admin% ssadm logdump -i local_log_file [output args] [filter args]
```

logdump fundamentally outputs either a stream of log records or readable text in various formats (after applying specified filters).

The presence of the -o option causes (binary) log records to be produced:

```
admin%ssadm logdump -i input_arg -o local_log_file [filter args]
```

Omit the -o option to output readable text.

The formatting options for readable text are common to snoop; these are -v, -V, -t [r|a|d], and -xoffset [,length]. For more information, see snoop(1M) man page.

logdump Extensions

logdump is an extension of the standard snoop packet monitoring tool provided with the Solaris operating environment. Any expertise in the use of snoop is directly applicable to use of logdump.

The facilities of logdump that are common to snoop are detailed in the ssadm-logdump(1m) man page.

logdump has been extended to provide for the special additional needs of SunScreen. These extensions are summarized as:

- Extensions to the format of network packets logged: inclusion of the interface on which logged packets arrive and inclusion of the reason packets are logged (whycodes)
- Provisions for SUMMARY logging (wherein only a size-limited packet preamble is logged)
- Logging of network packets on routing mode (ROUTING) interfaces removes the MAC-layer header
- Addition of session- and extended-log events (previously described)
- Enforcement and synthesis of unique timestamps
- True filter (pipeable) processing. Standard snoop can process a captured packet stream, or produce a captured packet stream, but not both; logdump allows both.
- Addition of filtering operators for selection of various log event types (loglvl), event severity (logsev), logging program component (logapp), packet log interface (logiface), and packet log reason (logwhy)
- Addition of range operands for IP addresses and port numbers to mirror the semantics of SunScreen address and service objects
- Display of SKIP packet encapsulations
- Display of all extensions listed above

logdump is also fundamentally different from snoop in that it is not involved in decisions as to what SunScreen logs. (Rules and variables previously described provide this control.) snoop is often used to filter network input during the process of capture or direct display. logdump serves as a means to postprocess log file content only.

SunScreen logs and snoop-captured files are not interoperable.

SunScreen welfmt Utility

The SunScreen welfmt utility reads a SunScreen binary log file and generates an ASCII log file in WebTrends Enhanced Log Format (WELF). This allows the WebTrends Firewall Suite (WFS) to analyze SunScreen log files and produce detailed reports on topics such as bandwidth usage, protocol distribution, email and Web activity, FTP transfers, and Telnet sessions.

WFS is a third-party product from WebTrends that provides a greater variety of visual log analysis tools. If it is already loaded on your system, ensure you are using Version 3.0 or later. For further details, see the WebTrends Web site. See also the ssadm-welfmt (1m) man page.

The following procedure, performed on a system with WFS installed, describes how to use the SunScreen welfmt utility to enable WFS to analyze your SunScreen log files. It also describes how to produce the WELF log file from a SunScreen binary log file.

Note – The welfmt utility is included on the SunScreen 3.2 CD-ROM.

1. Use the SunScreen configuration editor to retrieve the SunScreen log file from your Screen and save it:

```
# ssadm log get > binary-logfile
```

2. Run the welfmt utility with the output of step 1 as input and save the ASCII output:

```
# ssadm welfmt -f your-Screen-name -i binary-logfile > output-file
```

3. Import the welfmt output file to the system with the WFS product installed.

HTTP Proxy Header Logging

The proxies in SunScreen 3.2 produce a number of log events which are useful to the WebTrends Firewall Suite (WFS), (via the `welfmt` utility.) However, the default set of HTTP protocol header items logged by the HTTP proxy do not, by default, include several header items which extend the usefulness of the reports generated by WFS.

For backward-compatibility reasons, the default set of HTTP header items logged remains largely as in previous releases of SunScreen. To accommodate the needs of WFS users, the HTTP proxy has been augmented to enable additional header items to be mapped into its log events.

Configuration of this mapping facility is controlled by a pair of variables, one to control HTTP request headers, the other to control HTTP response headers. The requests header mappings are tailored by:

- `sys=Screen` (optional)
- `prg=http`
- `name=LogRqsts`
- `values={ hdrmaps }`
- `description="descriptive text"`
- `enabled | disabled` (as initially configured, it is disabled)

The response header mappings are tailored by:

- `sys=Screen` (optional)
- `prg=http`
- `name=LogRsps`
- `values={ hdrmaps }`
- `description="descriptive text"`
- `enabled | disabled` (as initially configured, it is disabled)

The `hdrmaps` portion is a list of one or more of the following:

- `group="WELF"`
- `name="namestr"`
- `name="!namestr"`
- `prefix="prefixstr"`
- `prefix="!prefixstr"`
- `suffix="suffixstr"`
- `suffix="!suffixstr"`

`group="WELF"` enables all headers useful to WFS (either request or response, depending upon the variable in which they appear). Other forms specify the full header names (`name=...`), header prefix strings (`prefix=...`), or header suffix strings (`suffix=...`) to map on an individual basis. Forms with an exclamation point (!) represent negating the sense of the match.

Mappings are processed in the order in which they occur in the `values={ ... }` list. Specific matches should appear before more general ones.

Enabling a mapping causes the header(s) which match to be promoted into the NOTICE log events, thus retaining their values in the SunScreen log.

Both variables -- `LogRqsts` and `LogRsps` -- are pre-defined. If you display their install-time values, you see:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=http name=LogRqsts
PRG="http" NAME="LogRqsts" DISABLED VALUES={ group="WELF" }
DESCRIPTION="request headers to log"
edit> vars print prg=http name=LogRsps
PRG="http" NAME="LogRsps" DISABLED VALUES={ group="WELF" }
DESCRIPTION="response headers to log"
```

Note that the variables must be changed to `ENABLED` to log headers optimal to WFS reporting.

Logged Network Packet Enhancements

SunScreen logs contain network traffic that arrives on multiple link-layer interfaces in a contemporarily interspersed manner. For this reason, it is important to record the interface upon which the traffic was received. The interface is noted by the name of its Solaris device (for example, `le0`, `elx0`).

Note – For `snoop`, the interface being monitored is specified as a command-line option. This information is not retained in a `snoop`-produced capture file.

Additionally, you can configure a Screen to log network traffic for a variety of reasons, such as packets that passed successfully, those that failed to match rules, those that arrived after session state expired, etc. The reason is recorded as an unsigned integer, commonly referred to as the why code. (See Appendix D for a complete table of these reasons.)

`logdump` displays these extended items and allows filtering based on these extended items as shown in the table below.

TABLE 11–3 Examples: Extended Items for logdump

Extended Items for logdump	Description
logiface <i>interface</i>	To restrict output based on an interface with the <code>logiface</code> operator. It takes as its argument the name (or name prefix) of the interface desired. The name is compared in a case-insensitive manner. For example, to restrict log events to network traffic arriving on any qe network device, type <code>ssadm -r Screen log get ssadm logdump -i- logiface qe</code> .
logwhy #	To restrict output based on the reason a packet was logged. The <code>logwhy</code> operator takes as its argument a number representing a reason code described above. For example, to restrict log events to network traffic that was passed and logged, type <code>ssadm -r Screen log get ssadm logdump -i- logwhy 1</code> .

General Event Type Enhancements

In addition to network packet traffic, logs can contain session summary events and extended log events. Each of these is represented by a different log record format.

Session summary events contain source and destination information regarding the session (for example, IP addresses and port numbers), plus ending statistics for the session. Extended log events are produced by various program components as previously described. `logdump` displays these new event types.

`logdump` allows discrimination of these types from network packet traffic events. The `loglvl` operator is provided to select network packet traffic, session summaries, authentication, and application events.

Log Record Format

The table below contains examples of the `logdump` filters that you can use to restrict the display of various events.

TABLE 11–4 Examples: Filters for Restriction Various events

Filter	Description
<code>loglvl pkt</code>	Restricts output to network packet traffic events. The <code>logiface</code> and <code>logwhy</code> operators imply <code>loglvl pkt</code> .

TABLE 11-4 Examples: Filters for Restriction Various events (Continued)

Filter	Description
loglvl sess	Restricts output to session summary events. In previous SunScreen releases, the <code>sas_logdump</code> command had <code>-S</code> and <code>-s</code> options that provided a crude form of the <code>loglvl sess</code> feature. Those options are no longer supported.
loglvl auth	Restricts output to authentication events.
loglvl app	Restricts output to application events.

The filtering mechanisms inherited from `snoop` related to IP addresses (for example, `host`, `to`, `from`, `dst`, `src`, and naked IP addresses and hostnames) have been extended to filter all event types that contain corresponding IP addresses. For example:

```
admin% ... ssadm log get from src_host > out_log
```

matches packet, session, and extended events that originated from the given source host.

Similarly, the filtering mechanisms inherited from `snoop` that are related to TCP and UDP ports (for example, `port`, `dstport` and `srcport`) have been extended to filter all event types that relate to the corresponding services. For example:

```
admin% ... ssadm log get port svc > out_log
```

matches packet, session, and extended events that relate to the given service.

Extended Log Event Enhancements

The extended events added to the SunScreen log contain additional fields as previously described (severity code and program component name). The extended log mechanism has been generalized to enable a wide variety of events to be recorded in the log. Because of the self-described syntax used, virtually any event can be added to the log in this manner.

`logdump` allows discrimination of extended events based on their severity code. The `logsev` operator provides this ability. The operand for `logsev` is one of the severity pseudonyms `emerg`, `alert`, `crit`, `err`, `warn`, `note`, `info`, or `debug`. These same designators are used to restrict the actual logging of these events. For example:

```
admin% ssadm -r Screen log get | ssadm logdump -i- logsev warn ...
```

matches extended events of a severity warning or greater.

logdump allows discrimination of extended events based on the name of the program component that logged them. The logapp operator performs this restriction. The operand for logapp is a string that is the name of a program component. For example:

```
admin% ssadm -r Screen log get | ssadm logdump -i- logapp ftpd ...
```

matches extended events for the FTP proxy.

Note – The logsev and logapp operators imply a filter of (loglvl auth or loglvl app).

All extended log events share some common optional attributes. These attributes are optional because they only occur in log events where they make sense. They are common in the sense that they are handled in a consistent way. These attributes are shown in the table below.

TABLE 11–5 Optional Attributes

Attribute	Description
sess_ID	A session serial number, used to recognize various events that are related to each other
proto_ip	IP protocol number (usually 6 for TCP or 17 for UDP)
src_ip	IP source address
src_port	IP source port number
dst_ip	IP destination address
dst_port	IP destination port number
reason	Short description of the event
msg	Generic message text

Log Filtering Macros

For SunScreen, log filters can be defined as named quantities. These are referred to as *log macros*.

Log macros are defined and stored in the registry (on the primary Screen) or they can be defined in a local registry on any Screen. Log macros defined in the registry of the

primary Screen are automatically propagated to secondary Screens in the same way as all other SunScreen registry objects. This propagation affords uniform log filter availability and ease of common usage across a collection of managed Screens.

Log macros can also be defined in a registry-like facility that is local to each secondary Screen. This local macro capability is provided for emergency situations and other cases where central macro definition and mass activation is unacceptable. Collect such local macros back into the central registry as soon as practical for permanent storage and propagation.

Log macros are named using a global- and Screen-specific two-level scheme similar to other objects in SunScreen. Evaluation mechanisms prefer a Screen-specific macro with a given name over a global one. Evaluation of macros occurs at the time of usage.

Note – If you are familiar with computer programming languages you will recognize this as a traditional delayed name-binding mechanism with dynamic scoping.

Log macros also provide a bridge between the namespace of address and service objects defined in the SunScreen registry and their potential usage (as resolved to values) by the filtering facilities of `logdump`.

Note – `logdump` filtering retains the host name-to-address and service name-to-port number mapping mechanisms of `snoop`—namely, DNS, NIS, host, and service tables defined for Solaris software.

Displaying and Creating Log Macros

Log macros are actually a derivative of the general SunScreen variable mechanism. Therefore, the variable naming and value structures exist for log macros, namely:

```
sys=Screenname (optional)
name=macroname
value="macrobody"
description="descriptive text" (optional)
enabled | disabled (default is enabled)
```

Log macros are configured in the registry using the `logmacro edit` subcommand of `ssadm`. For group-Screen installations, they are configured on the primary Screen.

Note – You do not have to save a log macro to use it; it is saved when it is created. However, to propagate log macro definitions from a primary Screen to secondaries, you must activate the configuration.

Examples: Log Macros on the Primary Screen

The following are examples of using log macros while logged into the primary Screen.

To display the definition of a non-Screen specific macro:

```
admin% ssadm -r primary edit Initial
edit> logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp" DESCRIPTION="SMTP mail" ...
```

To define a non-Screen specific macro:

```
admin% ssadm -r primary edit Initial
edit> logmacro add name=pkts-only value="loglvl pkt" Description="only network packets"
edit> quit
```

To define a macro for a specific Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro add sys=Screenname name=SFO value="port SFO" description="routing SFO"
edit> quit
```

Note – Although, the output produced by `print` surrounds the value of each item in double quotes, these are only necessary on *input* if there are embedded spaces within the values of items. Also, although `print` outputs all tag names in capital letters (for example, `NAME=`), these tags are recognized in a case-insensitive manner on input (for example, `name=`, `Name=`, `NAME=` are equivalent.)

You can create expediency log macros on any Screen using `logmacro` as a subcommand of `ssadm` (rather than an `ssadm edit` subcommand). The syntax of the rest of the usage is the same as given above.

Examples: Log Macros on the Secondary Screen

The following are examples of using log macros while logged into the secondary Screen.

To display the definition of a non-Screen-specific macro:

```
admin% ssadm -r secondary logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp"
```



```
DESCRIPTION="SMTP mail" ...
```

To define a macro for a specific Screen:

```
admin% ssadm -r secondary logmacro add sys=secondary  
name=SFO-routing value="port rip src SFO-routers"  
description="routing activity in SFO district"
```



Caution – Do not define log macros on secondary Screens which are not Screen-specific.

Log Macro Name and Body

The name of a log macro consists of a *name=macroname* part, preceded by an optional *sys=Screenname* Screen-restriction part.

Unlike many objects in SunScreen, the *macroname* portion must be formulated as a simple identifier rather than a more complicated general string. (A simple identifier begins with an ASCII alphabetic character or an underscore, followed by zero or ASCII alphanumeric characters or underscores.)

The *macrobody* (value part) of a log macro consists of a filtering expression suitable for logdump. In its simplest form, this is a string that can be used directly as filtering arguments.

However, the log macro expansion feature parses the value string looking for logdump operators that introduce address and service names and, finding same, attempts to resolve them from the SunScreen registry. So, for addresses, it looks for the operators *host*, *to*, *from*, *between*, *dst*, *src* and tries to resolve their operands in the address registry. If they are found, the operator-operand sequence is rewritten with the registry value for that address.

Similarly, for services, it looks for the operators *port*, *dstport*, and *srcport*. If their operand resolves in the service registry, the operator-operand sequence is rewritten with the registry value.

Note – In SunScreen, the registry services expanded in this manner can only consist of TCP or UDP services. Ranges of ports are allowed but groups are disallowed, as are services that use non-TCP non-UDP state engines.

Additionally, expansion looks for the operator *macro* and, if found, looks up the operand and replaces the operator-operand sequence with the named macro's body. Expansion cannot handle addresses or services from the registry that are not named with simple identifiers as well.

Listing Log Macros

Log macros in the primary registry can be displayed using the `logmacro` subcommand of `ssadm edit`. Individual macro definitions can be displayed. Also, all Screen-nonspecific definitions, or all definitions specific to a Screen, or all definitions specific to any Screen, can be displayed. You can generate an abbreviated listing that contains just the names of these last three classes of macros.

Examples: Macro Definitions for the Primary Screen

The following are examples of displaying definitions while logged into the primary Screen.

To display two specific macro definitions:

```
admin% ssadm -r primary edit Initial
edit> logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp" DESCRIPTION="SMTP mail" ...
edit> logmacro print sys=secondary name=SFO
SYS="secondary" NAME="SFO" VALUE="port SFO" DESCRIPTION="routing SFO" ...
```

To display all Screen-nonspecific definitions:

```
admin% ssadm -r primary edit Initial
edit> logmacro print
NAME="mail-only" ENABLED VALUE="svc smtp" DESCRIPTION="SMTP mail" ...
NAME="pkts-only" ENABLED VALUE="loglvl pkt" DESCRIPTION="only network packets" ...
```

To display all definitions specific to a Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro print sys=secondary
SYS="secondary" NAME="SFO" VALUE="port SFO" DESCRIPTION="routing SFO" ...
```

To display all definitions specific to any Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro print sys=
SYS="primary" NAME="HQ-routing" VALUE="port HQ-routers" DESCRIPTION="HQ routing" ...
SYS="secondary" NAME="SFO-routing" VALUE="port SFO" DESCRIPTION="routing SFO" ...
```

To display all Screen-nonspecific names:

```
admin% ssadm -r primary edit Initial
edit> logmacro names
NAME="mail-only"
NAME="pkts-only"
```

To display all names specific to a Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names sys=secondary
```

```
SYS="secondary" NAME="SFO-routing"
```

To display all names specific to any Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names sys=
SYS="primary" NAME="HQ-routing"
SYS="secondary" NAME="SFO-routing"
```

Examples: Macro Definitions for the Secondary Screen

The following are examples of displaying definitions while logged into the secondary Screen.

To display two specific macro definitions:

```
admin% ssadm -r secondary logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp" DESCRIPTION="SMTP mail" ...
admin% ssadm -r secondary logmacro print sys=secondary name=SFO-routing
SYS="secondary" NAME="SFO-routing" VALUE="port SFO" DESCRIPTION="routing SFO" ...
```

To display all Screen-nonspecific definitions:

```
admin% ssadm -r secondary logmacro print
NAME="mail-only" ENABLED VALUE="svc smtp" DESCRIPTION="SMTP mail" ...
NAME="pkts-only" ENABLED VALUE="loglvl pkt" DESCRIPTION="only network packets" ...
```

To display all definitions specific to a Screen:

```
admin% ssadm -r secondary logmacro print sys=secondary
SYS="secondary" NAME="SFO-routing" VALUE="port SFO" DESCRIPTION="routing SFO" ...
```

The following is an example of what you would type to produce name lists:

```
admin% ssadm -r secondary logmacro names sys=secondary
SYS="secondary" NAME="SFO-routing"
```

To display all Screen-nonspecific names:

```
admin% ssadm -r secondary logmacro names
NAME="mail-only"
NAME="pkts-only"
```

To display all names specific to a Screen:

```
admin% ssadm -r secondary logmacro names sys=secondary
SYS="secondary" NAME="SFO-routing"
```

Log Macro Usage

To use a log macro, you expand its value and cause that expansion to be presented as a filter expression to a `log get*` or `logdump` command.

To introduce examples of log macro expansion using `logmacro` as a subcommand to `ssadm`, the first example shows the defined values to two macros as rendered by `ssadm logmacro print`:

```
admin% ssadm -r Screen logmacro print
NAME="probed-ports" ENABLED VALUE="icmp or dstport telnet or dstport
  rlogin or dstport rsh or dstport ftp or srcport X11 or port adminweb"
admin% ssadm -r Screen logmacro print sys=
SYS="Screen" NAME="suspicious" ENABLED VALUE="logwhy 256 logiface le0
( not from trusted or to hidden ) macro probed-ports"
```

Two macros are defined. The first macro, `probed-ports`, is Screen-nonspecific and ostensibly defines services that are thought to be targets for initial probes leading to security attacks. The second macro, `suspicious`, is specific to a Screen and contains a more complete macro for filtering potential probes. It restricts itself to:

- Packets logged because no rule was found or that had source addresses that were illegal on their interface (`logwhy 256`)
- Packets arriving on a specific (presumably outside) interface (`logiface le0`)
- Packets originating from untrusted hosts or targeted at hosts that are unpublished (`not from trusted or to hidden`)
- Restrictions imposed by the macro "`probed-ports`"

Tip – The verb `names,flat` produces a list of names that are available for macro expansion on a particular Screen. For example, while logged into a Screen, type:

```
admin% ssadm -r Screen logmacro names,flat
"probed-ports"
"suspicious"
```

This hides Screen-specific issues of macros and lists macro names as they are used by embedded macro references.

Assume that the following definitions have been created and activated for registry items:

```
edit> list Address
"abraham" HOST 1.2.3.4
"hidden" RANGE 129.9.9..0 129.9.9.255
"john" HOST 2.3.4.5
"martin" HOST 3.4.5.6
"trusted" GROUP { "abraham" "martin" "john" } { }
edit> list Service
"rlogin" SIMPLE FORWARD "tcp" PORT 513
```

```
"rsh" SIMPLE FORWARD "tcp" PORT 514
"telnet" SIMPLE FORWARD "tcp" PORT 23
"X11" SIMPLE FORWARD "tcp" PORT 6000-6063
```

To expand a given macro, while logged into a Screen:

```
admin% ssadm -r Screen logmacro expand suspicious
logwhy 256 logiface le0 ( not ( from
1.2.3.4 or from 2.3.4.5 or from 3.4.5.6 ) or to
129.9.9.0..129.9.9.255 )
( icmp or dstport 23 or dstport 513 or dstport 514 or ( srcport
20 or dstport 21 ) or srcport 6000..6063 or port adminweb )
```

This usage illustrates various expansion and resolution operations performed by expand. The clause from trusted has been replaced by the registry values for the GROUP trusted. The clause to hidden has also been resolved to a registry RANGE, using the logdump syntax for IP address ranges a.b.c.d..e.f.g.h.

The embedded macro reference macro probed-ports has been expanded. The clauses that can be resolved from the registry (dstport telnet, dstport rlogin, dstport rsh, dstport ftp, and srcport X11) have been expanded using registry values. Clauses that were not found in the registry (icmp and port adminweb) were left to be resolved by logdump itself. The dstport ftp clause further illustrates some special processing employed for that protocol, and the expansion of the srcport X11 clause shows the logdump syntax for port ranges x..y.

Note – Resolution of SunScreen registry items performed by expand is made using those of the currently activated policy and for the Screen whereon the expand operation is executed.

The logmacro expand mechanism has been designed to facilitate simple command-line usage in conjunction with the other log processing facilities of SunScreen.

To employ the above macro to retrieve the suspicious items in the current log on the Screen and display them, while logged into the Screen:

```
admin% ssadm -r Screen log get 'ssadm -r Screen logmacro expand suspicious' |
ssadm logdump -v -i-
```


Migrating From Earlier SunScreen Firewall Products

The table below compares the commands used in SunScreen EFS, Release 2.0, and SunScreen SPF-200 to the equivalent commands used in SunScreen 3.2. These commands used in SunScreen EFS, Release 2.0, and SunScreen SPF-200 are obsolete in SunScreen 3.2 and are no longer supported.

Note – Typing the `ssadm edit` command invokes the configuration editor, which displays the `edit>` prompt. The “add” verb is used as an example, but other verbs such as “delete” or “list” can be used.

TABLE A-1 Command Compatibility Reference Table

Old Commands	SunScreen Equivalent
<code>gui_install</code>	<code>install</code>
<code>sas_logdump</code>	<code>ssadm logdump</code>
<code>sas_main</code>	Java GUI
<code>sas_registry</code>	Java GUI
<code>spf_admin_install</code>	<code>install</code>
<code>spf_upgrade_skip</code>	Not used
<code>ss_access</code>	<code>edit> add accessremote</code> <code>edit> add accesslocal</code>
<code>ss_action</code>	Not used
<code>ss_active_config</code>	<code>ssadm active</code>
<code>ss_address</code>	<code>edit> add address</code>
<code>ss_admin_user</code>	<code>edit> authuser add</code>

TABLE A-1 Command Compatibility Reference Table (Continued)

Old Commands	SunScreen Equivalent
ss_alerts	edit> add screen
ss_algorithm	ssadm algorithm
ss_authuser	edit> authuser add
ss_backup_all	ssadm backup
ss_certificate	edit> add certificate
ss_client <i>Screen ...</i>	ssadm -r <i>Screen ...</i>
ss_compile	ssadm activate
ss_compiler	Not used
ss_configuration	ssadm policy
ss_copy_key	Not used
ss_debug_level	ssadm debug_level
ss_default_drop	Not used
ss_defaultrouter	edit> add screen
ss_diff_config	Not used
ss_disable_send	Not used
ss_dns	edit> add screen
ss_do_compile	Not used
ss_domain	Not used
ss_editor	ssadm edit <i>policy</i>
ss_fix_multicast	Not used
ss_getskip	ssadm backup
ss_ha	ssadm ha
ss_ha_active_mode	ssadm ha active_mode
ss_ha_passive_mode	ssadm ha passive_mode
ss_ha_restart	Not used
ss_ha_status	ssadm ha status
ss_had	Not used
ss_init_interfaces	Not used

TABLE A-1 Command Compatibility Reference Table *(Continued)*

Old Commands	SunScreen Equivalent
ss_install	ssadm configure
ss_interfaces	edit> add Interfaces
ss_jar_hash	edit> jar_hash add
ss_jar_sig	edit> jar_sig add
ss_load_group	Not used
ss_log	ssadm log
ss_nat	edit> add nat
ss_network	edit> add screen
ss_patch	ssadm patch
ss_plumb_interface	Not used
ss_product	ssadm product
ss_restore_all	ssadm restore
ss_router	edit> add Interface
ss_rule	edit> add rule
ss_server	Not used
ss_service	edit> add service
ss_stateengine	edit> list stateengine
ss_sys_info	ssadm sys_info
ss_traffic_stats	ssadm traffic_stats
statetables -f	ssadm lib/statetables -f

Configuration Editor Reference

This appendix describes the command line interface and the configuration editor and the options available for each command. The following topics are included:

- “What Is the Configuration Editor?” on page 219
- “Solaris (shell) Commands” on page 221
- “ssadm” on page 221
- “ssadm Subcommands” on page 223
- “Unsupported Commands” on page 237
- “ssadm SKIP Commands” on page 239
- “Configuration Editor” on page 240
- “Network Monitoring and Maintenance” on page 272

The top-level programs are run directly from a Solaris shell prompt, if the standard administrative command directory, `/usr/sbin`, is included in your `$PATH`.

Other sections in this appendix describe the commands of the two major SunScreen subsystems that interpret commands: the `ssadm` subcommands and the configuration editor. There is also a section on unsupported commands that are obsolete or otherwise not guaranteed to exist in future releases of this product.

Note – If you are familiar with the Solaris shell and SunScreen concepts and terms, the command line is appropriate. If you are new to SunScreen, use the administration GUI to enter network settings.

What Is the Configuration Editor?

All the functionality of SunScreen that is available through the administration GUI is also available through the command line configuration editor. Administering your

Screens through the configuration editor is useful if you cannot or do not want to use the GUI. If you are adept with the command line, you can sometimes edit a configuration more quickly. You can also use the command line interface to perform management tasks in an automated fashion using scripts. Finally, the command line interface may be employed by users who require accessibility accommodations.

You can obtain access to a Screen using the command line from its own keyboard when the Screen is administered locally, if you have superuser (`root`) access. You can also gain access to a Screen using the command line from an Administration Station; when the Screen is administered remotely you must use SKIP or IKE encryption and an administration user name and password.

You maintain user-controlled data—common objects like address and policy entries like rules and NAT—using the `edit` subcommand of `ssadm`.

When using the configuration editor, you must usually save any changes before quitting. Some entities (`authuser`, `adminuser`, `proxyuser`, `logmacro`, `vars`) are saved automatically when created. See “Save Not Required for Some Common Objects” on page 220 below for more information.

You invoke the configuration editor with the `edit` command, a subcommand of `ssadm` and the name of your policy, such as `Initial`. The prompt for the configuration editor is `edit>`.

For a locally administered Screen, type:

```
# ssadm edit policy_name
```

For a remotely administered Screen, type:

```
# ssadm -r Screen_name edit policy_name
```

Save Not Required for Some Common Objects

You can quit the configuration editor without saving if only `authuser`, `adminuser`, `proxyuser`, `logmacro`, or `vars` entities are altered. The following is an example of the nonfatal message you see if you attempt to save after changing only the entities shown above:

```
edit> save
lock not held
failed (status 244)
```

Solaris (shell) Commands

The following Solaris (shell) commands are available at your shell prompt if the standard administrative command directory `/usr/sbin` is included in your `$PATH`.

- `ssadm`
- `ss_client`

The table below lists the SunScreen Solaris (shell) commands and their descriptions. Many of these commands duplicate administration GUI functions, while others provide a context for other commands.

TABLE B-1 Summary of SunScreen 3.2 Solaris (shell) Commands

Solaris Command	Description
<code>ss_client</code>	Provides communication between an Administration Station and a Screen that is running an earlier SunScreen firewall product release. <code>ss_client</code> is provided only for the purpose of remotely administering such products using the SunScreen system as a remote Administration Station.
<code>ssadm</code>	The primary command-line tool for SunScreen administration. <code>ssadm</code> subcommands perform various operations such as editing and activating a SunScreen configuration, and examining the status of a Screen.

`ssadm`

`ssadm` is the primary command-line tool for SunScreen administration. `ssadm` has a number of subcommands that perform various operations such as editing and activating a configuration, and examining the status of a Screen.

The Solaris command `ssadm` provides character-set translation between embedded strings and the local character set of the Solaris system on which it runs.

`ssadm` runs directly on a locally administered Screen, or indirectly from a remote Administration Station that is using SunScreen SKIP or IPsec/IKE to encrypt IP network communications passing between them. See the *SunScreen SKIP User's Guide, Release 1.5.1* for more information regarding SKIP encryption.

Usage:

`ssadm [-b] [-n] subcommand [parameters...]`

`ssadm [-b] [-n] -r remotehost [-F ticketfile] subcommand [parameters...]`

The table below describes the options for this command.

TABLE B-2 Options for `ssadm` Command

Options	Description
<code>-b</code> ¹	Processes binary data (instead of text) in standard input and output
<code>-n</code>	Does not read any input from standard input
<code>-r remotehost</code>	Provides access to a remote Screen using an address or a hostname <i>remotehost</i>
<code>-F ticketfile</code>	Use authorization ticket stored in <i>ticketfile</i>

1. The `-b` option is normally not needed because the commands that process binary data automatically enable the binary mode. For example, `ssadm backup`, `ssadm restore`, `ssadm log`, `ssadm logdump`, and `ssadm patch` handle binary data even if `-b` is not specified.

The available `ssadm` subcommands are each described in “`ssadm` Subcommands” on page 223.

When `ssadm` is executed locally on the Screen (that is, without the `-r` option) no login or authentication is required, but you must be superuser to have any effect.

When `ssadm` is used with the `-r` option to access a remote Screen, login authentication is required. You must use the `ssadm login` command to get a ticket that is used by subsequent invocations of `ssadm` to allow access to the remote Screen. Normally, the ticket is stored in a *ticketfile*, the name of which can be specified using the `-F` option, or through the `SSADM_TICKET_FILE` environment variable. See the `ssadm login` command for information about ticket files and remote administration using `ssadm`.

Executing an `ssadm` Command From a Local Screen

You can configure a local Screen by typing the commands listed in this appendix using the Screen’s keyboard. For example, to activate a policy named `Initial`, you type:

```
# ssadm activate Initial
```

The `ssadm` command resides in the `/usr/sbin` directory. Include this directory in your directory search path to have access to the commands on the local Screen.

Executing an `ssadm -r` Command on a Remote Administration Station

You can configure a Screen from a remote Administration Station by preceding the commands listed in this appendix with `ssadm -r` and the name of the Screen you want to administer. For example, to activate the policy named `Initial` on a remote Screen called `SunScreen1`, you type:

```
# ssadm -r SunScreen1 activate Initial
```

When `ssadm` is used with the `-r` option to access a remote Screen, the name of the *ticketfile* can be specified using the `-F` option, or through the `SSADM_TICKET_FILE` environment.

`ssadm` Subcommands

The following commands, which can be used as the subcommand argument to the `ssadm` command, are described in this section.

- `activate`
- `active`
- `algorithm`
- `backup`
- `certdb`
- `certlocal`
- `certrldb`
- `configure`
- `debug_level`
- `edit`
- `ha`
- `lock`
- `log`
- `logdump`
- `login`
- `logmacro`
- `logout`
- `logstats`
- `patch`
- `policy`
- `product`
- `restore`
- `skipca`
- `skipdb`

- skiplocal
- sys_info
- traffic_stats

ssadm Subcommand Summary

The table below lists the SunScreen `ssadm` subcommands and their descriptions. Many `ssadm` subcommands duplicate the functions of the administration graphical user interface, while others provide a context for other subcommands.

TABLE B-3 Summary of SunScreen `ssadm` Subcommands

<code>ssadm</code> Subcommand	Description
<code>activate</code>	Activates a policy on a Screen
<code>active</code>	Lists information about the currently active policy
<code>algorithm</code>	Lists algorithms supported by SKIP and IKE
<code>backup</code>	Writes a SunScreen backup file to standard output
<code>certdb</code>	Manages public key certificate databases
<code>certlocal</code>	Manages private key database
<code>certrldb</code>	Manages certificate revocation list database.
<code>configure</code>	Runs the text-based utility for creating the Initial SunScreen configuration (formerly <code>ss_install</code>)
<code>debug_level</code>	Sets or clears the level of debugging output generated by a Screen
<code>edit</code>	Runs the configuration editor. See “Configuration Editor” on page 240.
<code>ha</code>	Configures the features of a high availability (HA) Screen
<code>lock</code>	Examines or forces the removal of the protection lock that the configuration editor places on a policy file or the Registry file.
<code>log</code>	Maintains the Screen log file
<code>logdump</code>	Filters or displays log records, as retrieved by <code>ssadm log get</code>

TABLE B-3 Summary of SunScreen `ssadm` Subcommands (Continued)

<code>ssadm</code> Subcommand	Description
<code>login</code>	Authenticates a user for administrative access through <code>ssadm</code> to a Screen from a remote Administration Station
<code>logmacro</code>	Expands a SunScreen <code>logmacro</code> object
<code>logout</code>	Terminates the session created by <code>ssadm login</code>
<code>logstats</code>	Prints information about the SunScreen log
<code>patch</code>	Installs a patch, as needed
<code>policy</code>	Creates, deletes, lists and renames Screen policies
<code>product</code>	Prints a single-line description of SunScreen generic type
<code>restore</code>	Reads a backup file from standard input
<code>sys_info</code>	Prints a description of running SunScreen software
<code>traffic_stats</code>	Reports summary information about the traffic flowing through the SunScreen, classified by interface

`ssadm activate`

`ssadm activate` causes the Screen to begin “executing” a particular configuration that is formed when the named policy is combined with the common objects. After activation, the configuration controls the behavior of packet filtering, encryption and decryption, proxies, logging, and administrative access.

Usage:

```
ssadm activate [-n] [-l] policy
```

The table below describes the options for this command.

TABLE B-4 Options for `activate` Subcommand

Options	Description
<code>-n</code>	Verifies that configuration is valid; does not make it active

TABLE B-4 Options for activate Subcommand (Continued)

Options	Description
-l	Does not send the configuration to other Screens in the centralized management group, only activates it on the local Screen.

The named *policy* is combined with the common objects to form a configuration.

Note – If you omit the *policy* argument, `ssadm activate` reads a configuration file from standard input. Since the format of a configuration file is undocumented and private to the SunScreen internal software, using `ssadm activate` in this way is not supported.

ssadm active

`ssadm active` prints a description of the configuration that is currently being *executed* by the Screen. When run with the `-x` option, the configuration file is extracted from the running system and can be saved for later examination.

Usage:

```
ssadm active
```

```
ssadm active -x policy
```

Without the `-x` option, `ssadm active` describes the active configuration with two lines of text. The first line lists the name of the Screen on which the configuration was originally stored, the name of the internal database in which it was stored (this name is always `default`), and the name of the policy, including its version number. The second line lists the date and time when the configuration was activated, and the user (either a Solaris user or SunScreen administration authorized user) who caused it to be activated.

For example:

```
# ssadm active
Active configuration: greatwall default Initial.3
Activated by admin on 03/09/1999 02:58:36 PM PST
```

In this example, the Screen is currently running a configuration that came from the Screen named `greatwall` (which might be the current Screen or, if the Screen is a member of a centralized management group, the primary Screen of the centralized administration group). The configuration includes version 3 of the policy `Initial`.

With the `-x` option, `ssadm active` saves the active configuration into the named *policy* that can be examined using the `edit` command. The named *policy* must not

already exist; `ssadm active` creates the policy. The `-x` option differs from a normal policy. A normal policy file contains only policy rules, which are combined with the currently-defined common objects. The policy saved by the `-x` option contains a full snapshot of all common objects, in addition to the policy rules.

Note – If the `-x` option is specified and the *policy* argument is omitted, `ssadm active` writes a configuration file to standard output. Since the format of a configuration file is undocumented and private to the SunScreen internal software, using `ssadm active` in this way is not supported.

ssadm algorithm

`ssadm algorithm` lists the SKIP and IKE algorithms that are available for a specified algorithm type.

Usage:

```
ssadm algorithm [-i] alg_type[crypt_type]
```

OR

```
ssadm algorithm [-k] alg_type[crypt_type]
```

where *alg_type* must be one of key, data, mac, compression, encryption, or authentication, and *crypt_type*, if supplied, must be SKIP_VERSION_1, SKIP_VERSION_2, or IPSEC.

The following combinations are valid:

```
ssadm algorithm i/k-opt key SKIP_VERSION_1
ssadm algorithm i/k-opt data SKIP_VERSION_1
ssadm algorithm i/k-opt key SKIP_VERSION_2
ssadm algorithm i/k-opt data SKIP_VERSION_2
ssadm algorithm i/k-opt mac SKIP_VERSION_2
ssadm algorithm i/k-opt compression SKIP_VERSION_2
ssadm algorithm i/k-opt encryption [IPSEC]
ssadm algorithm i/k-opt authentication [IPSEC]
```

The *i/k-opt* is either `-i`, or `-k`. Note that `-i` lists only algorithms (with the specified restrictions) that are currently installed on the Screen and that `-k` lists all possible (known) algorithms (with the specified restrictions). The default is `-k`.

As shown above, the default *crypt_type* for key, data, mac, and compression is SKIP_VERSION_2; the default *crypt_type* for encryption and authentication is IPSEC.

ssadm backup

ssadm backup writes a Screen backup file to standard output.

Usage:

```
ssadm backup [-v] > file
```

The backup file contains the complete configuration of SKIP and IKE, plus all currently defined common objects, policies, and, if the `-v` option is specified, all of the saved versions of the policies.

The backup file can be restored at a later time using the `ssadm restore` command.



Caution – SECURITY WARNING. The file created by `ssadm backup` contains sensitive information (SKIP and IKE secret keys) that must be stored and disposed of appropriately to protect the integrity of the Screen.

ssadm certdb

ssadm certdb allows a user to manually administer the two databases of public key certificates used by IKE and SKIP. These databases store long term certificates so that they may be accessed by the key manager.

Usage:

```
ssadm certdb -[I|S] -[a|e|h|l|r] [action specific arguments]
```

where `-I` or `-S` instructs the command to access the IKE or SKIP database and `a`, `e`, `h`, `l`, and `r` represent add, extract, help, list, or remove. See the man page `ssadm-certdb(1M)` for details.

Note – The semantics and applicability of the options may vary between IKE and SKIP usage. For SKIP options, see the `skipdb` man page

ssadm certlocal

ssadm certlocal is a utility for managing the two local identity databases used by IKE and SKIP on a system.

Usage:

```
ssadm certlocal -[I|S] -[a|e|h|k|l|r] [action specific arguments]
```

where `-I` or `-S` instructs the command to access the IKE or SKIP database and `a`, `e`, `h`, `l`, and `r` represent add, extract, help, generate key, list, or remove. See the man page `ssadm-certlocal(1M)` for details.

Note – The semantics and applicability of the options may vary between IKE and SKIP usage. For SKIP options, see the `skiplocal` man page

ssadm certrldb

`ssadm certrldb` is a utility for managing the certificate revocations lists in the IKE certificate database.

Usage:

```
ssadm certrldb -[I] -[a|e|h|l|r] [action specific arguments]
```

where `-I` instructs the command to access the IKE database and `a`, `e`, `h`, `l`, and `r` represent add, extract, help, list, or remove. See the man page `ssadm-certrldb(1M)` for details.

ssadm configure

`ssadm configure` (formerly `ss_install`) is a text-based command-line utility that runs during SunScreen installation to create an initial configuration. `ssadm configure`, combined with `pkgadd`, is the command-line equivalent to the installation wizard graphical user interface on the CD-ROM.

`ssadm configure` interactively queries you with various configuration options. It then creates a configuration, stores it under the policy name `Initial`, and activates it.

After `ssadm configure` is complete, the Screen is ready to be administered using the administration GUI or the command-line configuration editor and other tools.

ssadm debug_level

`ssadm debug_level` controls the output of internal debugging information from the SunScreen kernel.

Usage:

```
ssadm debug_level [newlevel]
```

`ssadm debug_level ?`

With no arguments, `ssadm debug_level` prints out the current debug level in hexadecimal. With the *newlevel* argument, `ssadm debug_level` sets the debug level to *newlevel*. With the question mark argument (may need to be quoted in the Solaris shell), `ssadm debug_level` prints out a list of bit values and their meanings.

The debugging information, when enabled, is written through the kernel message mechanism and typically ends up on the system console or the kernel message logs. The format of the messages is not documented and is only used by Sun support personnel.

ssadm edit

`ssadm edit` runs the SunScreen configuration editor.

Usage:

`ssadm edit policy`

`ssadm edit policy < file`

`ssadm edit policy -c commandstring`

See “Configuration Editor” on page 240 for information regarding commands supported by `ssadm edit`. The configuration editor can be used in any of three modes: interactive, batch, or “-c” mode. In interactive mode, the editor prints a prompt (`edit>`) before each command is read from your terminal. In batch mode, the editor silently reads commands from standard input. Commands are read until the editor receives end-of-file or a `quit` command.

If `ssadm edit` is run on an interactive terminal and its input and output are not redirected, it automatically enters interactive mode. If standard input is a pipe or a file, the configuration editor runs in batch mode.

If `ssadm edit` is run with the `-c` option, it executes the *commandstring* and then exits without reading any other commands. *commandstring* must be a single argument to the program, so in the Solaris shell it usually has to be quoted with single or double quotes.

ssadm ha

`ssadm ha` performs operations on a Screen in a high availability (HA) cluster.

Usage:

`ssadm ha` *function parameters...*

The table below describes the function parameters for this command.

TABLE B-5 Function Parameters for ha Subcommand

Functions	Descriptions
<code>status</code>	Displays status of the HA cluster.
<code>active_mode</code>	Puts the Screen in active mode.
<code>passive_mode</code>	Puts the Screen in passive mode.
<code>init_primary interface</code>	Turns a standalone (non-HA) Screen into an HA primary Screen, thereby creating a new HA cluster containing one Screen. <i>interface</i> is the interface to be used for the HA heartbeat and synchronization. <i>primaryIP</i> is the IP address (on the HA network) of the primary machine in the cluster.
<code>init_secondary interface primaryIP</code>	Turns a standalone (non-HA) Screen into an HA secondary screen ready to join an existing HA cluster. <i>interface</i> is used for the HA heartbeat and synchronization, and <i>primaryIP</i> is the IP address (on the HA network) of the primary machine in the cluster.
<code>add_secondary secondaryIP</code>	Adds an initialized HA secondary Screen (<i>see init_secondary</i> above) into an existing HA cluster. This command is executed on the primary Screen in the HA cluster. <i>secondaryIP</i> is the IP address (on the HA network) of the secondary machine to be added.

ssadm lock

`ssadm lock` manipulates the lock that protects a policy from simultaneous modification by multiple administrators.

Usage:

`ssadm lock -w policy`

`ssadm lock -c policy`

`ssadm lock -w` prints a line of text describing the status of the lock.

`ssadm lock -c` forcibly breaks the lock and attempts to terminate (with a SIGHUP signal) the previous holder of the lock.

For example:

```
# ssadm lock -w Initial
Lock held by admin@198.41.0.6 process id:8977
# ssadm lock -c Initial
# ssadm lock -w Initial
Lock available
```

ssadm log

ssadm log retrieves and clears the SunScreen log.

Usage:

```
ssadm log get filter_args...
ssadm log get_and_clear filter_args...
ssadm log clear
```

See Chapter 11 for detailed information.

ssadm logdump

ssadm logdump is used to filter or display log records, as retrieved by ssadm log get.

Usage:

```
ssadm logdump parameters...
```

See Chapter 11 for detailed information.

ssadm login

ssadm login authenticates a user for administrative access through ssadm to a Screen from a remote Administration Station.

Usage:

```
ssadm -r remotehost login username password
```

ssadm login creates a session on the remote Screen and provides a *ticket* that allows subsequent invocations of the ssadm command to access the remote Screen without using a password.

`ssadm login` is only available with the `-r remotehost` option.

The ticket is written to standard output. If a *ticketfile* is specified using the `-F` option to `ssadm` or the `SSADM_TICKET_FILE` environment variable, then `ssadm login` automatically stores the ticket in *ticketfile* in addition to writing it to standard output.

For example:

```
# SSADM_TICKET_FILE=$HOME/.ssadmticket
# export SSADM_TICKET_FILE
# touch $SSADM_TICKET_FILE
# chmod go= $SSADM_TICKET_FILE
# ssadm -r greatwall login admin password
WRITE access <E23B344150C702EC>
# ssadm -r greatwall activate Initial
Configuration activated successfully on greatwall.
# ssadm -r greatwall active
Active configuration: greatwall default Initial.3
Activated by admin on 03/09/1999 02:58:36 PM PST
# ssadm -r greatwall logout
```

The above example is for `sh` or `ksh`; other shells may require different commands. `ssadm login` is only available with the `-r remotehost` option.

When using the `ssadm login` command on multiuser Administration Stations, any other user can snoop the admin user and password using `ps`, then (because SKIP or IKE is enabled from that host) access the Screen as that user.



Caution – Do not have a general-use Solaris system act as a remote Administration Station. Additionally, *never* use the `ssadm login` command on a Solaris system while other users are logged in

Screen administration is discouraged from non-Solaris platforms. Serious security holes with other operating systems can readily be exploited to compromise the network security infrastructure.

See the `ssadm-login(1M)` man page for more information on the `login` command.

ssadm logout

`ssadm logout` terminates the session created by `ssadm login`.

Usage:

```
ssadm -r remotehost logout
```

`ssadm logout` is only available with the `-r remotehost` option.

ssadm logmacro

ssadm logmacro expands a SunScreen logmacro object.

Usage:

```
ssadm logmacro expand macroname
logmacro add macrokey macrovalue
logmacro delete macrokey
logmacro print[,sortopt] [ macrokey ]
logmacro names[,sortopt]
```

where macrokey is of the form [SYS=*scrnname*] NAME=*name* macrovalue is of the form VALUE=*macrobody* sortopt is one of asc, desc, iasc

(For example, desc specifies a plaintext description string desc to be associated with the object.

See Chapter 11 for detailed information.

ssadm logstats

ssadm logstats prints information about the SunScreen log.

Usage:

```
ssadm logstats
```

ssadm patch

ssadm patch installs a patch, as needed.

Usage:

For stealth-mode Screens from Remote Administration Stations, use:

```
ssadm [-r screen_name] patch Install [NOREBOOT] < patch.tar.Z
ssadm [-r screen_name] patch Backout [NOREBOOT] patchID
```

On routing-mode Screens, the standard Solaris patchadd and patchrm commands can be used.

If a SunScreen software patch is needed, detailed instructions are provided with the patch.

ssadm policy

ssadm policy creates, deletes, renames, or lists the defined policies.

Usage:

```
ssadm policy -a policies...
ssadm policy -c oldname newname
ssadm policy -d [-v] policies...
ssadm policy -l [-v] [policies...]
ssadm policy -r oldname newname
```

The table below describes the options for this command.

TABLE B-6 Options for policy Subcommand

Options	Description
-a	Creates policies with the specified names. The newly created policies contain no rules and reference the currently defined common objects.
-c	Creates a policy named <i>newname</i> as a copy of the existing policy named <i>oldname</i> .
-d	Deletes the named policies. The specified <i>policies</i> can be either generic policy names, such as <i>Initial</i> , or specific versions, such as <i>Initial.3</i> . When a generic policy name is specified and the <i>-v</i> option is specified, <i>ssadm policy -d</i> deletes all of the versions of the policy. When a specific version is specified, only that version is deleted.
-l	Lists the named <i>policies</i> (or all policies available if no <i>policies</i> are given). The <i>-v</i> option also lists all of the saved versions of the policies.
-r	Renames the existing policy <i>oldname</i> to <i>newname</i> .

ssadm product

ssadm product prints out a single line of text describing the SunScreen product in use.

Usage:

```
ssadm product
```

ssadm restore

ssadm restore reads a backup file from standard input. The backup file must have been created using the backup command.

Usage:

```
ssadm restore < file
```

ssadm spf2efs

ssadm spf2efs converts a set of configuration data saved from a SunScreen SPF-200 Screen into SunScreen format.

Usage:

```
ssadm spf2efs < file
```

ssadm sys_info

ssadm sys_info prints a description of the running SunScreen software.

Usage:

```
ssadm sys_info
```

For example:

```
# ssadm sys_info
Product:      SunScreen
Version:      Release 3.1, March 10 2000(v0310991418)
System Boot Time:    03/15/1999 03:51 PST
SunScreen Boot Time:  Mon Mar 13 03:51:56 PST 200
```

traffic_stats Subcommand

`ssadm traffic_stats` reports summary information about the traffic flowing through the Screen, classified by interface.

Usage:

```
ssadm traffic_stats interfaces...
```

Unsupported Commands

Commands listed in this section are only used for abnormal maintenance or customer support functions, or as a temporary workaround to limitations of the current software.

These commands are “unstable,” which means that they may not be provided in future SunScreen product releases, or in versions for other operating systems.

```
ssadm lib/nattables
```

```
ssadm lib/screeninfo
```

```
ssadm lib/statetables
```

```
ssadm lib/support
```

```
ssadm SKIP commands
```

ssadm lib/nattables

`ssadm lib/nattables` lists the contents of internal NAT tables.

ssadm lib/screeninfo

`ssadm lib/screeninfo` runs in sequence several of the functions of the `ssadm lib/screeninfo` command, printing out a large set of information about the Screen and its current configuration.

Usage:

```
ssadm lib/screeninfo
```

The output of this command can be redirected to a file and may be requested by Sun’s Support services if you encounter problems with your Screen.

ssadm lib/statetables -f

ssadm lib/statetables -f causes the Screen to flush (discard) all of its connection state information. This causes all previously active connections through the Screen to be effectively disconnected.

The -f option is often useful after activating a modified policy that disallows some traffic that was previously allowed. Without running statetables -f, you allow any previously existing connections to remain active even if the new policy does not allow them. Running statetables -f causes all previously existing state sessions to be disconnecte; the active policy applies to any subsequent connections.

The -fs or -f -s option sets all IKE security associations (SAs) that are in kernel SADB to “expired” by setting their lifetime to the current time. The expired SAs can be renegotiated if they are needed. This option does not apply to IPsec manual SAs. Manually-keyed SAs never expire.

ssadm lib/support

ssadm lib/support provides various diagnostic and status information that can be useful when requesting customer support for the SunScreen product.

Usage:

ssadm lib/support *function parameters...*

This information may be requested by Enterprise Services if you encounter problems with your Screen.

Note – If you have any support issues, call your authorized service provider. For further information about support, use the following URL to contact Sun’s Support services: <http://www.sun.com/service/support/index.html>.

The major functions are shown in the table below.

TABLE B-7 Support Command Functions

Functions	Description
config	Bring over configuration files for the active policy

TABLE B-7 Support Command Functions (Continued)

Functions	Description
date	Set and get current time/date (SET DATE WITH CAUTION!)
disks	Check disk space (df -k)
eeprom	Check eeprom settings
findcore	Check if a core file exists
help	Prints a listing of functions available for this command
last	Check boot history (last)
packages	Check pkginfo and patch history
procs	Check processes (ps -elf)
skip	Check contents of /etc/skip/ directory
stats	Check the kernel networking statistics (netstat -k)
streams	Check the STREAMS statistics (netstat -m)
versions	Bring over version information on major SunScreen components

ss_client

ss_client is equivalent to the command of the same name provided with earlier SunScreen firewall products, such as *SunScreen EFS, Release 2.0*, or *SunScreen SPF-200*. ss_client is provided only for the purpose of remotely administering such products using the SunScreen system as a remote Administration Station.

Usage: ss_client *hostname command*

For information on how to use ss_client to administer an earlier SunScreen firewall product, see the documentation for that product.

ssadm SKIP Commands

```
ssadm lib/skipca
```

```
ssadm lib/skipd_restart
```

```
ssadm lib/skipdb
```

```
ssadm lib/skiplocal
```

```
ssadm lib/skipstat
```

These commands provide access to the command-line administration tools of the SunScreen SKIP software. They are primarily useful when a Screen is not physically accessible or does not allow logging in over the network and you want to configure SKIP using the `ssadm` command with the `-r` option from a remote Administration Station.

See the corresponding commands in the *SunScreen SKIP User's Guide, Release 1.5.1* for information on the parameters.

Configuration Editor

The configuration editor is the primary command-line tool for creating and manipulating the objects that control the operation of a Screen.

Configuration Editor Data Model

The table below lists the data types that compose the Data Model as maintained by the configuration editor (`ssadm edit`) and the `ssadm policy` command.

TABLE B-8 Configuration Editor Object Type Names

Object Type Name	Storage	Access Method	Description
address	common	named	Addresses of network elements
screen	common	named	Screen objects and their relationships
state engine	common (read only)	named	Filtering capabilities of packet filter engine.
service	common	named	Network services that can be filtered
interface	common	named	Network interfaces of a Screen.
certificate	common	named	Certificate used for SKIP or IKE connections

TABLE B-8 Configuration Editor Object Type Names *(Continued)*

Object Type Name	Storage	Access Method	Description
key	common	named	Manual encryption keys for IPsec manual mode and pre-shared keys for IKE usage
time	common	named	Time intervals for time-dependent rules
authuser	external	named	Users for administration and/or proxy access
proxyuser	external	named	Users for proxy access
jar_hash	external	named	Java archive hash (for HTTP proxy applet filtering)
jar_sig	external	named	Java archive signature (for HTTP proxy applet filtering)
logmacro	external	named	Log filtering macro definitions
mail_relay	external	named	Mail relays (for SMTP proxy mail filtering)
mail_spam	external	named	Spam domains (for SMTP proxy mail filtering)
policy	policy list	named	Multiple named policies for storing different configurations
filter rule	policy	ordered	Network traffic filtering/control/logging
nat rule	policy	ordered	NAT translations
local access rule	policy	ordered	Who can access the Screen for local administration and what they can do
remote access rule	policy	ordered	Who can access the Screen for remote administration and what they can do
vars	external	named	General environment-like configuration variables

TABLE B-8 Configuration Editor Object Type Names *(Continued)*

Object Type Name	Storage	Access Method	Description
VPN gateway	policy	ordered	Define which hosts (addresses) are protected by which Screens, and the encryption mechanisms to be employed
VPN			All vpngateway objects with the same name constitute / define (virtually) any given VPN. That name is used in filter rules, causing the VPN to use the encryption mechanisms of the vpngateway

Object types marked as having common storage in the table are normally stored in the common objects registry that is not part of any individual policy. These objects are used by all policies, so changes to the common objects can affect the behavior of multiple policies. To edit the common objects, specify a policy name when starting the configuration editor even if you are not modifying any policy objects.

Object types marked as having policy storage in Table B-8 are stored as part of a policy. Policy objects often refer to common objects and, therefore, can behave differently depending on the value of common objects. For example, a policy can contain a rule object that allows address A to communicate with address B. The address objects A and B are defined in the common objects.

Object types marked as having external storage in the table are almost equivalent to common objects, but they are stored in a separate database that is not affected by the `quit`, `reload`, or `save` commands. Changes to these objects are always stored immediately, and persist even if the `savecommand` is not used.

Object types marked as having policy list storage in Table B-8 represent the names of the policies themselves. A policy currently being edited can be saved or cloned (or portions of it) into a new policy. Other policy requests, such as `add`, `delete`, and `rename` are provided by the `ssadm policy` command.

Configuration Editor Subcommands

The `ssadm edit` commands are used when running the configuration editor, which is responsible for maintaining the SunScreen configuration database.

The table below lists the SunScreen configuration editor `ssadm edit` subcommands and their descriptions. Many subcommands duplicate administration GUI functions, while others provide a context for other subcommands.

TABLE B-9 SunScreen Configuration Editor `ssadm edit` Subcommands

<code>edit</code> Subcommand	Description
<code>add</code>	Create or redefine an entry
<code>add_member</code>	Add member to an Address, Certificate, Key, or Service group
<code>authuser</code>	Manipulate the list of authorized users
<code>del [ete]</code>	Delete the specified entry of the given TYPE
<code>del [ete]_member</code>	Delete a member from an Address, Certificate, Key, or Service group
<code>insert</code>	Insert a new object of one of the ordered (indexed) types in a specified position in the corresponding list
<code>jar_hash</code>	Manipulate the list of JAR hashes used by the HTTP proxy
<code>jar_sig</code>	Manipulate the list of JAR signatures used by the HTTP proxy
<code>list</code>	Display all data for all entries or a specific entry of a given TYPE
<code>list_name</code>	Display the set of unique basenames and subtypes of all of a given TYPE
<code>load</code>	Load a policy into the configuration editor
<code>lock</code>	Lock the Registry and policy in anticipation of performing edits
<code>lock_status</code>	Return the status of the lock relative to this editor
<code>mail_relay</code>	Manipulate the list of mail relays used by the SMTP proxy
<code>mail_spam</code>	Manipulate the list of spam domains used by the SMTP proxy
<code>move</code>	Move an indexed entry from its current location in the ordered list to the new location
<code>proxyuser</code>	Manipulate the list of proxy users

TABLE B-9 SunScreen Configuration Editor `ssadm edit` Subcommands (Continued)

<code>edit</code> Subcommand	Description
<code>refer</code>	Determine if a named object of a given TYPE is referred to in the Registry or the current policy
<code>referlist</code>	Display a list of all entries in the Registry or the current policy that refer to a specified named-object of a given TYPE
<code>reload</code>	Discard any and all edits, if made, and reload the data into the editor from the database.
<code>rename</code>	Rename a specified named-object of a given TYPE
<code>renamereference</code>	Rename all references to a specified named-object of a given TYPE
<code>replace</code>	Replace an object at a specified index
<code>save</code>	Save all current edits to the Registry and policy
<code>saveas</code>	Save the data currently in the editor under new name
<code>search</code>	Search the Registry for objects that match specified criteria
<code>vars</code>	Manipulate general configuration variables
<code>verify</code>	Takes no arguments and verifies the currently loaded policy
<code>quit</code>	Cause the editor to terminate if there are no unsaved changes
<code>QUIT</code>	Cause the editor to terminate even if there are unsaved changes

In the following command descriptions, *name_TYPE* indicates that it requires the name of an object of a particular TYPE. A pound sign (#) indicates that it needs an index. <KEYWORD> now indicates a keyword that previous SunScreen releases required and that now is optional.

add

Creates or redefines an entry.

Usage:

add TYPE *parameters...*

If a named-type is specified and an entry with that name already exists, it is replaced with the new entry. If it does not exist, one is created with the new name. All of the following have a similar request of `add_nocheck`, which does not perform consistency checking.

add address

```
add address "name_ADDRESS" <HOST> #.#.#.#
```

```
add address "name_ADDRESS" <RANGE> #.#.#.# #.#.#.#
```

```
add address "name_ADDRESS" #.#.#.# - #.#.#.#
```

```
add address "name_ADDRESS" #.#.#.#/#.#.#.#
```

```
add address "name_ADDRESS" #.#.#.#/#bits
```

```
add address "add address "name_ADDRESS" <GROUP> { "name_ADDRESS"
... } { "name_ADDRESS" ... }
```

The following fields are optional and can be specified in any order *after* the address keyword:

```
SCREEN "name_SCREEN"
```

```
COMMENT "comment string"
```

Note – The addresses `*` and `localhost` are reserved and cannot be edited.

add screen

```
add screen "name_SCREEN"
```

The following fields are optional and can be specified in any order *after* the screen keyword:

```
MASTER "name_SCREEN"
```

```
HA_PRIMARY
```

```
HA_SECONDARY
```

```
TIMEOUT #
```

```

SNMP #.#.#.# ... (list can be empty; not output if empty list)
SNMP_TIMER # (if SNMP is set)
CDP {"on" if present, "off" otherwise}
RIP {"on" if present, "off" otherwise}
DNS {"on" if present, "off" otherwise}
NIS {"on" if present, "off" otherwise}
LOGSIZE # {default is 100 MBytes if not present}
DEST_CHECK {destination address checking}
STEALTH_NET #.#.#.# #.#.#.# {Network and Netmask for stealth type
Interfaces}
STEALTH_NET #.#.#.#/#.#.#.#
STEALTH_NET #.#.#.#/#bits
HA_IP #.#.#.# (required if HA_PRIMARY is set)
HA_ETHER xx:xx:xx:xx:xx:xx (required if HA_PRIMARY is set)
COMMENT "comment string"

```

If the Screen is to be a CMG slave Screen, the following SKIP and/or IKE fields must be specified as well. They can be specified in any order *after* the SCREEN keyword. The SKIP fields are:

```

ADMIN_IP #.#.#.# or name_ADDRESS
ADMIN_CERTIFICATE "name_CERTIFICATE"
KEY "name_key_algorithm"
DATA "name_data_algorithm"
MAC "name_mac_algorithm"
COMPRESSION "name_compression_algorithm"
TUNNEL "name_address"

```

The IKE fields are:

```

ADMIN_IP #.#.#.# or "name_ADDRESS"
AH ( "name_auth_algorithm" )

```

ESP ("name_encr_algorithm")

ESP ("name_encr_algorithm", "name_auth_algorithm")

Note – At least one of the above *must* be present. At most, one of the ESP forms *can* be present.

IKE ("name_encr_algorithm", "name_auth_algorithm", "oakley_group_#",
name_auth_method", name_CERTIFICATE")

Note – If both SKIP and IKE CMG are in use, only one instance of ADMIN_IP is allowed (or needed).

If the Screen is to be a CMG master Screen, the following SKIP and/or IKE fields must be specified as well. They can be specified in any order *after* the SCREEN keyword. The SKIP fields are:

ADMIN_IP #.#.#.# or "name_ADDRESS"

ADMIN_CERTIFICATE "name_CERTIFICATE"

The IKE fields are:

ADMIN_IP #.#.#.# or "name_ADDRESS"

IKE ("name_CERTIFICATE")

Note – If both SKIP and IKE CMG are in use, only one instance of ADMIN_IP is allowed (or needed).

The screen * is reserved and cannot be edited.

add service

add service "name_SERVICE" <SINGLE> filter ...

add service "name_SERVICE" GROUP "name_SERVICE" ...

For SINGLE services, a list of Filters follows the SINGLE keyword. The list must not be empty. Each Discriminator list also must not be empty. A Filter is of the form:

FORWARD "name_STATEENGINE" discriminator ...

REVERSE "*name_STATEENGINE*" *discriminator* ...

An individual discriminator is as follows:

PORT #

PORT #-# (No space is allowed before or after the - character)

BROADCAST #

BROADCAST #-#

An optional parameter for discriminators, which appears immediately *after* discriminator number or range it modifies, is:

PARAMETERS *space-separated list of #*

For GROUP services, a space-separated list of "*name_SERVICE*" entries follows the GROUP keyword.

The following fields are optional and can be specified in any order *after* the service keyword:

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

The service * is reserved and cannot be edited.

add interface

add interface "*name_INTERFACE*" type "*name_ADDRESS*"

type must be one of ADMIN, DISABLED, ROUTING, HA, or STEALTH.

The following fields are optional for stealth interface types and can be specified in any order *after* the interface keyword. Up to five ROUTERs per stealth interface can be specified. More can be specified, but only five are used by the system. The system may use the five stealth interfaces randomly in any order

ROUTER #.#.#.#

The following fields are optional for *all* interface types and can be specified in any order *after* the "interface" keyword:

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

The following fields are optional for *all* interface types *except* DISABLED and can be specified in any order *after* the interface keyword.

LOG NONE {default if no LOG is specified}

LOG SUMMARY

LOG DETAIL

ICMP NONE

ICMP NET_UNREACHABLE

ICMP HOST_UNREACHABLE

ICMP PORT_UNREACHABLE

ICMP NET_FORBIDDEN

ICMP HOST_FORBIDDEN

SNMP {"on" if present, "off" otherwise}

add certificate

```
add certificate "name_CERTIFICATE" SINGLE NSID # MKID "#"
```

```
add certificate "name_CERTIFICATE" SINGLE IKE "ike certspec" ...
```

```
add certificate "name_CERTIFICATE" GROUP "name_CERTIFICATE" ...
```

```
add certificate "name_CERTIFICATE" { "name_CERTIFICATE" ... }
```

```
add certificate "name_CERTIFICATE" { "name_CERTIFICATE" ... } " {  
"name_CERTIFICATE" }
```

For GROUP certificates, a space-separated list of *name_CERTIFICATE* entries is given in the first pair of braces (or after the *GROUP* keyword).

For IKE certificate groups, a list of *name_CERTIFICATE* entries may also be given in the second pair of braces. Like the Address object, this second list represents certificates (or criteria) which are to be excluded. Unlike Address group objects, only a top-level Certificate group may have a non-empty exclusion list.

Note – Groups which intermix SKIP and IKE Certificates are not allowed.

The following field is optional for SINGLE entries and may be specified in any order *after* the `certificate` keyword:

LOCAL "*name_SCREEN*"

The following fields are optional and can be specified in any order *after* the `certificate` keyword:

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

add key

add key "*name_KEY*" SINGLE *hexadecimal key value*

Note – Key objects exist in the same namespace as Certificate objects. Therefore, you cannot use the same name for both. Keys can also have SCREEN and COMMENT option fields.

add time

add time "*name_TIME*"

The following fields are optional and can be specified in any order *after* the `time` keyword:

EVERYDAY

SUNDAY

MONDAY

TUESDAY

WEDNESDAY

THURSDAY

FRIDAY

SATURDAY

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

The time object * cannot be modified.

Following any of the *DAY keywords can be a time of day specification in the form {timespec ...}. *timespec* is a time range in the form:

Start Hour:Start Minute Stop Hour:Stop Minute

Examples are: { 1:00 2:30 } and { 1:00 2:30 4:00 6:00 }. Twenty-four-hour time format is used, so the valid times are 0:00 (starting at midnight) through 24:00 (ending at midnight).

add rule

add rule "*name_SERVICE*" name_ADDRESS

Appends the rule to the end of the list of rules in the policy. `insert rule` should be used to position a new rule into an existing policy.

The following fields are optional and can be specified in any order *after* the rule keyword:

ALLOW {default if no ACTION specified}

DENY

LOG NONE {also LOG_NONE, default if no LOG is specified}

LOG SUMMARY {also LOG_SUMMARY}

LOG DETAIL {also LOG_DETAIL}

LOG SESSION {also LOG_SESSION, only valid for ALLOW rules, will be error for DENY}

SNMP {"on" if present, "off" otherwise}

USER "*name_USER*" {required only if PROXY_FTP or PROXY_Telnet set below; optional if 'PROXY_HTTP' set below; otherwise not allowed}

TIME "*name_TIME*"

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

Any one of the following combo-fields is optional and only valid in a rule that has ALLOW specified. It can be specified anywhere *after* the rule keyword:

```
SKIP_VERSION_1 "name_CERTIFICATE" "name_CERTIFICATE"  
"name_KEY_ALGORITHM" "name_DATA_ALGORITHM"
```

```
SKIP_VERSION_2 "name_CERTIFICATE" "name_CERTIFICATE"  
"name_KEY_ALGORITHM" "name_DATA_ALGORITHM"  
"name_MAC_ALGORITHM" "name_COMPRESSION_ALGORITHM"
```

```
IPSEC SYMMETRIC AH (ah_spi_value, "name_AUTHENTICATION_ALGORITHM",  
"name_KEY")
```

```
IPSEC SYMMETRIC AH (ah_spi_value "name_AUTHENTICATION_ALGORITHM",  
"name_KEY") ESP ( esp_spi_value, "name_ENCRYPTION_ALGORITHM", "name_KEY")
```

```
IPSEC SYMMETRIC ESP ( esp_spi_value, "name_ENCRYPTION_ALGORITHM",  
"name_KEY", name_AUTHENTICATION_ALGORITHM, "name_KEY")
```

```
IPSEC FORWARD AH (ah_spi_value, "name_AUTHENTICATION_ALGORITHM",  
"name_KEY") ESP ( esp_spi_value, "name_ENCRYPTION_ALGORITHM", "name_KEY")  
REVERSE AH (ah_spi_value, "name_AUTHENTICATION_ALGORITHM", "name_KEY")  
ESP ( esp_spi_value, "name_ENCRYPTION_ALGORITHM", "name_KEY")
```

```
IPSEC IKE ( "name_ENCRYPTION_ALGORITHM",  
name_AUTHENTICATION_ALGORITHM, OAKLEY_GROUP,  
"name_AUTHENTICATION_METHOD", "name_CERTIFICATE", name_CERTIFICATE")
```

```
IPSEC IKE ( "name_ENCRYPTION_ALGORITHM",  
name_AUTHENTICATION_ALGORITHM, OAKLEY_GROUP, PRE-SHARED,  
"name_KEY")
```

The first three IPsec symmetric forms (those with SPI values) specify manual keying. The asymmetric manual key form uses forward and reverse directions with AH and ESP specified separately for each direction. The last two IPsec forms utilize Internet Key Exchange (IKE) keying. Of those, the first form uses certificates, the last uses pre-shared keying. For either of the IKE forms, one of the following three data security parameter options (phase 2 transforms) must be specified. It may be issued *after* the IPSEC keyword:

```
AH ( "name_AUTHENTICATION_ALGORITHM" )
```

```
AH ( "name_AUTHENTICATION_ALGORITHM" ) ESP (  
"name_ENCRYPTION_ALGORITHM" )
```

```
ESP (  
"name_ENCRYPTION_ALGORITHM", name_AUTHENTICATION_ALGORITHM" )
```

The following fields are optional and only valid within a SKIP_VERSION_1, SKIP_VERSION_2, or IPSEC combo-field. They can be specified in any order *after* the combo-field:

SOURCE_TUNNEL "name_ADDRESS"

DESTINATION_TUNNEL "name_ADDRESS"

One or both of the following fields must be specified in conjunction with either IPsec manual keying or IKE pre-shared keying. They indicate to the SunScreen compiler the (encryption) role being played by a given Screen. They can be specified in any order *after* the IPSEC combo-field. Tip: when in doubt, completely specify both Screen roles:

SOURCE_SCREEN name_SCREEN

DESTINATION_SCREEN name_SCREEN

For IKE with certified keying material, the Screen roles are determined automatically, by determining which certificate (source or destination) is local to the Screen for which a policy is being compiled. If both source and destination certificates are (or contain) local entities, the *_SCREEN option may be used to disambiguate roles.

The following field is optional and only valid in a rule that has DENY specified. It can be specified anywhere *after* the rule keyword:

ICMP NONE {also ICMP_NONE, default if nothing is specified}

ICMP NET_UNREACHABLE {also ICMP_NET_UNREACHABLE}

ICMP HOST_UNREACHABLE {also ICMP_HOST_UNREACHABLE}

ICMP PORT_UNREACHABLE {also ICMP_PORT_UNREACHABLE}

ICMP NET_FORBIDDEN {also ICMP_NET_FORBIDDEN}

ICMP HOST_FORBIDDEN {also ICMP_HOST_FORBIDDEN}

The following field is optional and only valid in a rule that has ALLOW specified and no SKIP, IKE, IPsec, or proxy information. It can be specified anywhere *after* the rule keyword:

VPN "name_VPN"

The following fields are optional and only valid in a rule that has not specified any SKIP, IKE, or IPsec information and no VPN. They can be specified anywhere *after* the rule keyword. Only one of them can be specified in a given rule.

PROXY_FTP

PROXY_HTTP

PROXY_SMTp

PROXY_Telnet

The following fields are optional and only valid in a rule that has specified PROXY_FTP. They can be specified anywhere *after* the PROXY_FTP keyword:

FTP_GET

NO_FTP_GET {default if FTP_GET not specified}

FTP_PUT

NO_FTP_PUT {default if FTP_PUT not specified}

FTP_CHDIR

NO_FTP_CHDIR {default if FTP_CHDIR not specified}

FTP_MKDIR

NO_FTP_MKDIR {default if FTP_MKDIR not specified}

FTP_RENAME

NO_FTP_RENAME {default if FTP_RENAME not specified}

FTP_REMOVE_DIR

NO_FTP_REMOVE_DIR {default if FTP_REMOVE_DIR not specified}

FTP_DELETE

NO_FTP_DELETE {default if FTP_DELETE not specified}

FTP_ALL {same as FTP_GET FTP_PUT FTP_CHDIR FTP_MKDIR FTP_RENAME
FTP_REMOVE_DIR FTP_DELETE}

NO_FTP_ALL {default if no FTP options are present}

The following fields are optional and only valid in a rule that has specified PROXY_HTTP. They can be specified anywhere *after* the PROXY_HTTP keyword:

COOKIES

NO_COOKIES {default if COOKIES not specified}

ACTIVE_X

NO_ACTIVE_X {default if ACTIVE_X not specified}

SSL

NO_SSL {default if SSL not specified}

JAVA_SIGNATURE

JAVA_HASH

JAVA_SIGNATURE_HASH

JAVA

NO_JAVA {default if no other JAVA setting is specified}

HTTP_ALL {same as ACTIVE_X COOKIES JAVA SSL}

NO_HTTP_ALL {default if no HTTP options are present}

The following fields are optional and only valid in a rule that has specified PROXY_SMTP. They can be specified anywhere after the PROXY_SMTP keyword:
RELAY

NO_RELAY {default if RELAY not specified}

add nat

In the following two subcommands, *name_ADDRESS* has four different meanings: the first is source, the second is destination, the third is translated source, and the fourth is translated destination.

```
add nat STATIC "name_ADDRESS" "name_ADDRESS" "name_ADDRESS"  
"name_ADDRESS"
```

```
add nat DYNAMIC "name_ADDRESS" "name_ADDRESS" "name_ADDRESS"  
"name_ADDRESS"
```

The following fields are optional and can be specified in any order after the nat keyword:

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

add accesslocal

```
add accesslocal USER "name_USER"
```

The following fields are optional and can be specified in any order after the `accesslocal` keyword:

```
SCREEN "name_SCREEN"
```

```
COMMENT "comment string"
```

add accessremote

```
add accessremote USER "name_USER" "name_ADDRESS" SKIP_VERSION_1  
"name_CERTIFICATE" "name_KEY_ALGORITHM" "name_DATA_ALGORITHM"
```

```
add accessremote USER "name_USER" "name_ADDRESS" SKIP_VERSION_2  
"name_CERTIFICATE" "name_KEY_ALGORITHM" "name_DATA_ALGORITHM"  
"name_MAC_ALGORITHM" "name_COMPRESSION_ALGORITHM"
```

```
add accessremote USER "name_USER" "name_ADDRESS" IPSEC IKE (  
"name_ENCRYPTION_ALGORITHM", "name_AUTHENTICATION_ALGORITHM",  
OAKLEY_GROUP, "name_AUTHENTICATION_METHOD", "name_CERTIFICATE" )
```

For the IKE form, one of the following three data security parameter options (phase 2 transforms) must be specified. It may be issued *after* the IPSEC keyword:

```
AH ( "name_AUTHENTICATION_ALGORITHM" )
```

```
AH ( "name_AUTHENTICATION_ALGORITHM" ) ESP (  
"name_ENCRYPTION_ALGORITHM" )
```

```
ESP (  
"name_ENCRYPTION_ALGORITHM", "name_AUTHENTICATION_ALGORITHM" )
```

The following field is optional for `accessremote` entries. It can be specified in any order *after* the `accessremote` keyword:

```
TUNNEL "name_ADDRESS" { if the remote machine is using tunneling }
```

The following fields are optional and can be specified in any order after the `accesslocal/accessremote` keyword:

```
PERMISSION ALL
```

```
PERMISSION WRITE
```

```
PERMISSION READ
```

```
PERMISSION STATUS
```

```
PERMISSION NONE { default if no PERMISSION is specified }
```


SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

add vpngateway

add vpngateway "*name_VPN*" "*name_ADDRESS*" SKIP "*name_CERTIFICATE*"

add vpngateway "*name_VPN*" "*name_ADDRESS*" IPSEC IKE (
"*name_ENCRYPTION_ALGORITHM*", *name_AUTHENTICATION_ALGORITHM*,
OAKLEY_GROUP, "*name_AUTHENTICATION_METHOD*", "*name_CERTIFICATE*")

For the IKE form, one of the following three data security parameter options (phase 2 transforms) must be specified. It may be issued *after* the IPSEC keyword:

AH ("*name_AUTHENTICATION_ALGORITHM*")

AH ("*name_AUTHENTICATION_ALGORITHM*") ESP (
"*name_ENCRYPTION_ALGORITHM*")

ESP (
"*name_ENCRYPTION_ALGORITHM*", *name_AUTHENTICATION_ALGORITHM*")

For the SKIP form the following fields are required and can be specified in any order *after* the vpngateway keyword:

KEY "*name_KEY_ALGORITHM*"

DATA "*name_DATA_ALGORITHM*"

MAC "*name_MAC_ALGORITHM*"

COMPRESSION "*name_COMPRESSION_ALGORITHM*"

All vpngateway entries with the same name should have exactly the same encryption parameter settings, except for *name_CERTIFICATE*.

The following fields are optional and can be specified in any order *after* the vpngateway keyword:

TUNNEL "*name_ADDRESS*"

COMMENT "*comment string*"

add_member

Adds a member to a group or list.

Usage:

```
add_member address "name_ADDRESS" "name_ADDRESS"* { add to include list }
```

```
add_member address "name_ADDRESS" EXCLUDE "name_ADDRESS"* { add to exclude list }
```

```
add_member service "name_SERVICE" "name_SERVICE"*
```

```
add_member certificate "name_CERTIFICATE" "name_CERTIFICATE"* { add to include list }
```

```
add_member certificate "name_CERTIFICATE" EXCLUDE "name_CERTIFICATE"*  
{ add to exclude list; certificate groups can have exclude lists, which behave  
syntactically like address groups}
```

Note – The * denotes that multiple space-separated names can be specified in a single request.

The following field may be necessary to uniquely identify an entry. If so, you can specify it after the TYPE keyword:

```
SCREEN "name_SCREEN"
```

authuser

Manipulates the list of authorized users.

Usage:

```
authuser add name parameters...
```

```
authuser delete name
```

```
authuser print
```

```
authuser names
```

See “Authorized User” on page 130 for detailed information.

delete

Deletes the specified entry of the given TYPE.

Usage:

```
del [ete] address "name_ADDRESS"
*del [ete] screen "name_SUNSCREEN"
del [ete] service "name_SERVICE"
del [ete] interface "name_INTERFACE"
del [ete] certificate "name_CERTIFICATE"
del [ete] time "name_TIME"
*del [ete] rule #
*del [ete] nat #
*del [ete] accesslocal #
*del [ete] accessremote #
*del [ete] vpngateway #
*del [ete] key name_KEY
```

The following field may be necessary to uniquely identify an entry. If so it can be specified *after* the TYPE keyword, except for the entries preceded by an * above:

```
SCREEN "name_SCREEN"
```

delete_member

Deletes a member from a group or list.

Usage:

```
del [ete]_member address "name_ADDRESS" "name_ADDRESS" * { from include
list }

del [ete]_member address "name_ADDRESS" EXCLUDE "name_ADDRESS" *
{ from exclude list }

del [ete]_member service "name_SERVICE" "name_SERVICE" *
```

```
del[ete]_member certificate "name_CERTIFICATE" "name_CERTIFICATE" * {  
from include list }
```

```
del[ete]_member certificate "name_CERTIFICATE"  
EXCLUDE "name_CERTIFICATE" { from exclude list }
```

Note – * denotes that multiple space-separated names can be specified in a single request.

The following field may be necessary to uniquely identify an entry. If so it can be specified *after* the TYPE keyword:

```
SCREEN "name_SCREEN"
```

insert

Inserts a new object of one of the ordered (indexed) types in a specified position in the corresponding list.

Usage:

```
insert rule # parameters...
```

```
insert nat # parameters...
```

```
insert accesslocal # parameters...
```

```
insert accessremote # parameters...
```

```
insert vpngateway # parameters...
```

Index indicates the position the new entry holds in the list *after* it is inserted. The same syntax used for add is used for insert, with the index coming immediately *after* the TYPE keyword.

jar_hash

Manipulates the list of JAR hashes used by the HTTP proxy.

Usage:

```
jar_hash add name hash
```

```
jar_hash del name
```

```
jar_hash rename oldname newname
```

```
jar_hash list
```

```
jar_hash list_names
```

The table below describes the functions for this command.

TABLE B-10 Functions for jar_hash Subcommand

Functions	Description
add	Adds an entry to the jar_hash database.
del	Deletes an entry from the jar_hash database.
list	Lists the entries in the jar_hash database.
list_names	Lists the names of the entries in the jar_hash database
rename	Rename an entry in the jar_hash database

jar_sig

Manipulates the list of JAR signatures used by the HTTP proxy.

Usage:

```
jar_sig add name sig-hash
```

```
jar_sig del name
```

```
jar_sig rename oldname newname
```

```
jar_sig list
```

```
jar_sig list_names
```

The table below describes the functions for this command.

TABLE B-11 Functions for jar_sig Subcommand

Functions	Description
add	Adds an entry to the jar_sig database.
del	Deletes an entry from the jar_sig database.
list	Lists the entries in the jar_sig database.
list_names	Lists the names of the entries in the jar_sig database

TABLE B-11 Functions for jar_sig Subcommand (Continued)

Functions	Description
rename	Renames an entry in the jar_sig database

list

Displays all data for all entries or a specific entry of a given TYPE. The format of the output is the same as the syntax of the corresponding Add TYPE request.

Usage:

```
*list address
list address "name_ADDRESS"

*list key
list keyname_KEY

*list screen
*list screen "name_SCREEN"

*list service
list service "name_SERVICE"

*list stateengine
*list stateengine "name_STATEENGINE"

*list interface
list interface "name_INTERFACE"

*list certificate
list certificate "name_CERTIFICATE"

*list time
list time "name_TIME"

*list rule
*list rule #

*list nat
```

```
*list nat #  
  
*list accesslocal  
  
*list accesslocal #  
  
*list accessremote  
  
*list accessremote #  
  
*list vpngateway  
  
*list vpngateway #
```

The SCREEN "*name_SCREEN*" field is optional and can be specified after the TYPE keyword in any of the above requests except those that are preceded by an asterisk (*).

If no SCREEN option is present, only entries not associated with a specific SCREEN are listed. If the SCREEN option value is * , then all entries that otherwise match are displayed. Requests that do not specify a name always display all entries of the given type.

list_name

Displays the set of unique basenames and subtype of all of a given *TYPE*. These are the values that can be used when another object refers to an object of the specified *TYPE*.

Usage:

```
list_name TYPE
```

TYPE can be any of address, screen, service, state engine, interface, certificate, key, time, or vpngateway.

load

Loads a policy into the configuration editor.

Usage:

```
load "name_POLICY"
```

Any edits to the current policy must be saved or discarded before this operation will succeed.

lock

M

Manipulates the lock that protects a policy from simultaneous modification by multiple administrators.

Usage:

```
ssadm lock -w policy
```

```
ssadm lock -c policy
```

ssadm lock -w prints a line of text describing the status of the lock.

ssadm lock -c forcibly breaks the lock and attempts to terminate (with a SIGHUP signal) the previous holder of the lock.

For example:

```
# ssadm lock -w Initial
Lock held by admin@198.41.0.6 process id:8977
# ssadm lock -c Initial
# ssadm lock -w Initial
Lock available
```

lock_status

Returns the status of the lock relative to this editor. If this editor holds a lock, the type of lock is returned. If it does not hold a lock, another process acquired a WRITE lock. If that WRITE lock is still in effect, information about that WRITER is presented. If that WRITE lock is no longer in effect, then lock available is returned.

Usage:

```
lock_status
```

search

Searches the registry for objects that match specified criteria.

Usage:

```
search TYPE [SCREEN "name_SCREEN"] [ SUBTYPE subtype ] <EXACT>
Substring...
```


TYPE can be any of address, screen, service, state engine, interface, certificate, key, or time. *SUBTYPE* values depend upon the *TYPE* being searched according to the table below..

TABLE B-12 Search *TYPE*

TYPE	SUBTYPE
address	HOST, RANGE, GROUP
certificate	SINGLE, GROUP
key	
certificate	SINGLE, GROUP
interface	ADMIN, DISABLED, ROUTING, HA, STEALTH
screen	
service	SINGLE, GROUP
stateengine	
time	

The EXACT keyword requires a substring be specified and will only match entries whose name is an exact match.

move

Moves an indexed entry from its current location in the ordered list to the new location.

Usage:

move rule # #

move nat # #

move accesslocal # #

move accessremote # #

move vpngateway # #

replace

Replaces an object at a specified index.

Usage:

```
replace rule # parameters...
```

```
replace nat # parameters...
```

```
replace accesslocal # parameters...
```

```
replace accessremote # parameters...
```

```
replace vpngateway # parameters...
```

replace is similar to insert, except it replaces the entry at the specified index. It is shorthand for an insert *n* / del *n*+1 pair of requests. The same syntax used for add is used for replace, with the index coming immediately *after* the TYPE keyword.

refer

Determines if a named object of a given TYPE is referred to in the common data or the current policy.

Usage:

```
refer address "name_ADDRESS"
```

```
refer key "name_KEY"
```

```
refer screen "name_SCREEN"
```

```
refer service "name_SERVICE"
```

```
refer stateengine "name_STATEENGINE"
```

```
refer certificate "name_CERTIFICATE"
```

```
refer time "name_TIME"
```

```
refer vpn "name_VPN"
```

referlist

Displays a list of all entries in the common objects and/or the current policy that refer to a specified named-object of a given TYPE.

Usage:

```
referlist address "name_ADDRESS"
referlist screen "name_SCREEN"
referlist key "name_KEY"
referlist service "name_SERVICE"
referlist stateengine "name_STATEENGINE"
referlist certificate "name_CERTIFICATE"
referlist time "name_TIME"
referlist vpn "name_VPN"
```

rename

Renames a specified named object of a given TYPE.

Usage:

```
rename address "name_ADDRESS" "name_ADDRESS"
rename key "name_KEY" "name_KEY"
*rename screen "name_SCREEN" "name_SCREEN"
rename service "name_SERVICE" "name_SERVICE"
rename interface "name_INTERFACE" "name_INTERFACE"
rename certificate "name_CERTIFICATE" "name_CERTIFICATE"
rename time "name_TIME" "name_TIME"
```

The following field may be necessary to uniquely identify an entry. If so it can be specified *after* the TYPE keyword except for the entries preceded by an * above:

```
SCREEN "name_SCREEN"
```

If an entry already exists with the new name, it is replaced by this operation.

renamereference

Renames all references to a specified named object of a given TYPE.

Usage:

```
renamereference address "name_ADDRESS" "name_ADDRESS"
renamereference key "name_KEY" "name_KEY"
renamereference screen "name_SCREEN" "name_SCREEN"
renamereference service "name_SERVICE" "name_SERVICE"
renamereference certificate "name_CERTIFICATE" "name_CERTIFICATE"
renamereference time "name_TIME" "name_TIME"
renamereference vpn "name_VPN" "name_VPN"
```

save

Saves any current edits. It also creates a new version of the policy.

Usage:

```
save
save "name_POLICY"
```

If a name is specified, the current policy is written to the new name, but it remains the policy in the editor until you activate the policy written to the new name.

saveas

Saves the data currently in the editor under a new name.

Usage:

```
saveas policy "name_POLICY"
saveas policy "name_POLICY" registry Registry
saveas policy "name_POLICY" registry "name_POLICY"
saveas policy registry Registry
```

These are the only supported legal requests. The first saves the current policy data under the new name with a reference to the current common objects. The second saves the policy data under the new name and overwrites the current common objects with whatever is currently in the editor. This may affect other policies and should be done with caution. The third saves the contents of the editor, policy objects and common objects into a single self-contained policy file of the specified name. The last saves the common objects currently in the editor as the current common objects. This may affect other policies and should be done with caution.

reload

Discards any and all edits (if any were made) and reloads the data into the editor from the database. If another process has performed edits and saved them since the current editor process loaded its data, those edits will be lost if a `reload` is not performed before the current editor process makes further edits.

Usage:

```
reload
```

verify

Takes no arguments and verifies the currently loaded policy.

Usage:

```
verify
```

mail_relay

Manipulates the list of allowed and disallowed destination domains used by the SMTP proxy.

Usage:

```
mail_relay add relay_domain
```

```
mail_relay add !relay_domain
```

```
mail_relay del relay_domain
```

```
mail_relay del !relay_domain
```

`mail_relay list`

`add` adds a domain suffix string to be allowed (or disallowed, if preceded by a `!`) in recipients of mail messages.. `del` deletes a domain suffix string. `list` produces a list of the current set of relay domain suffixes.

mail_spam

Manipulates the list of "spam" domains or addresses used by the SMTP proxy.

Usage:

`mail_spam add spam_domain`

`mail_spam add IPaddress`

`mail_spam add IPstartaddress..IPendaddress`

`mail_spam del spam_domain`

`mail_spam del IPaddress`

`mail_spam del IPstartaddress..IPendaddress`

`mail_spam list`

`add` adds a domain suffix string to be blocked as an origin of incoming mail messages. `add` can also be configured to use an IP address or range of addresses; this blocks incoming messages from the addressed hosts lacking registered domain names. `del` removes a `spam_domain` suffix or IP address spam restrictor. `list` produces a list of the current set of spam restrictors.

proxyuser

Manipulates the list of proxy users.

Usage:

`proxyuser add name parameters...`

`proxyuser delete name`

`proxyuser print`

See "Proxy Users" on page 135 for detailed information.

vars

Manipulates general-purpose SunScreen variable objects.

Usage:

```
vars add varkey varvalue
```

```
vars del varkey
```

```
vars print [,sortopt] [varkey]
```

```
vars names [,sortopt]
```

varkey is of the form:

```
[ SYS=scrnname ]
```

```
[ PRG=prgname [ PGRP=pgrpname ] ]
```

```
NAME=name
```

quit

Causes the editor to terminate if there are no unsaved changes.

Usage:

```
quit
```

When the editor is used interactively, typing `quit` twice consecutively causes the editor to terminate even if there are unsaved changes.

QUIT

QUIT (typed in upper case) causes the editor to terminate even if there are unsaved changes.

Usage:

```
QUIT
```

Network Monitoring and Maintenance

The following sections describe how to monitor and maintain your SunScreen.

Using the `ssadm logdump` Command

`ssadm logdump` is based on the Solaris `snoop` program and has similar characteristics. In addition to the packet information available with `snoop`, SunScreen's logging mechanisms add information such as the interface on which the packet was received and the reason that the packet was logged. Any filtering language operation that works in `snoop` will work in `logdump`.

For details about `ssadm logdump`, see Chapter 11 and the `ssadm-logdump` man page.

To run `ssadm logdump` and display packets in a saved log file:

```
# ssadm logdump -i logfile
```

Where *log_file* is a log file that is downloaded from the Screen.

Note – Except for the differences detailed in Chapter 11, `logdump` uses the same filter language as the `snoop (1m)` program. Note also that `logdump` does not handle IPv6.

Using the `ssadm debug_level` Command

If you have access to the console on your SunScreen (through a serial line or directly connected keyboard and display), you can use the `ssadm debug_level` command to control the printing of command debugging information from the SunScreen kernel.

Typing `ssadm debug_level` with no arguments displays the current debug-level mask. By default, this mask is 1, which means it only reports significant errors.

If you specify a hex number as an argument for `ssadm debug_level`, the kernel debugging mask is set to that level. To get a list of debugging bit choices, type:

```
# ssadm debug_level ?
```

You select a `ssadm debug_level` mask by setting all of the debugging bits in which you are interested.

Probably the most useful of the `ssadm debug_level` debugging bit is `DEFAULT_DROP`. For example, if you type:

```
# ssadm debug_level 1001
```

any packets being dropped by SunScreen because they do not match any rule are reported. This is a quick way to see if the SunScreen is passing packets that you expect it to pass. You can also achieve this same result by setting the default action on the interface to `LOG_SUMMARY` or `LOG_DETAIL` and examine the logs.

Another useful debugging bit to set is `STATE_CHANGE`. This causes the kernel to report any additions or deletions from its internal state tables.

Some of the debugging bits produce a very large amount of output on a production Screen and should be used with caution. An example is `ACTION`, which reports execution of any PFL action.

TIP: it is often useful to employ a pair of `ssadm debug_level` commands, separated by the Solaris `sleep(1)` command, especially for levels which generate large amounts of output:

```
# ssadm debug_level
Current debug_level is: 00000001<>
# ssadm debug_level 1c01 ; sleep 30 ; ssadm debug_level 1
```

This would ensure that only 30 seconds of debug would be logged. This also avoids the mistake of leaving debugging enabled by accident.

Gathering Information From Your System to Report Support Issues

If you have any support issues, call your authorized service provider. For further information about support, use the following URL to contact Enterprise Services: <http://www.sun.com/service/support/index.html>.

For the most efficient help, first gather information describing your configuration. This information can be collected by saving the output of the following SunScreen support command. You invoke these commands to gather information that is useful in troubleshooting through the `ssadm lib/support` command.

The support command has the form: `ssadm [-r Screen_Name] lib/support function parameters...`

See “Unsupported Commands” on page 237.

Services and State Engines

This appendix describes the standard services, network service groups, and state engines supported by SunScreen. The following topics are included:

- “Standard Services” on page 275
- “* Service” on page 282
- “Network Service Groups” on page 292
- “State Engines” on page 295

Standard Services

SunScreen is shipped with a number of predefined network services. The table below lists the services in SunScreen, along with the state engine and discriminator (port, RPC program number, or type) for each service. Parameters (state engine modifiers, such as time-outs) and BROADCAST are indicated where applicable.

Service information is stored in the common object registry. See “add service” on page 247 in Appendix B.

Note – Some of the services in the table are described at the end of this table. The * service is not included in the table, but is described in “* Service” on page 282.

TABLE C–1 SunScreen Services

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
ah	iptunnel	IP protocol 51		

TABLE C-1 SunScreen Services (Continued)

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
archie	udp	port 1525 parameters (360 -1 0)		
auth	tcp	port 113		
automount	pmap_tcp pmap_udp rpc_tcp rpc_udp	program no. 300019 program no. 300019 program no. 300019 program no. 300019		
Backweb	udp	port 370 parameters (60 0 3)		
biff	udp_datagram	port 512 (BROADCAST)		
bootp	udp	port 67 (BROADCAST) parameters (60 0 3)		
certificate discovery	udp	port 1640 parameters (60 1 1)		
chargen	tcp	port 19		
CoolTalk	tcp udp_datagram	ports 6499-6500	port udp_datagram	port 13000 13000
CU See Me	udp_datagram	ports 7648-7652		
daytime	tcp	port 13		
daytime-udp	udp	port 13		
discard	tcp	port 9		
discard-udp	udp	port 9		
dns	tcp	port 53		
	dns	port 53		
echo	tcp	port 7		
echo-udp	udp	port 7		
esp	iptunnel	IP protocol 50		
exec	tcp	program no. 512		
finger	tcp	port 79		
ftp	ftp	port 21		

TABLE C-1 SunScreen Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
gopher	tcp	port 70		
HA	tcp	port 3856		
HA administration	tcp	port 3856		
HA heartbeat	ping	port 8		
icmp all	icmp	*		
icmp echo-reply	icmp	type 0		
icmp echo-request	icmp	type 8		
icmp exceeded	icmp	type 11		
icmp info	icmp	types 13 14 15 16 17 18		
icmp params	icmp	type 12		
icmp quench	icmp	type 4		
icmp redirect	icmp	type 5		
icmp unreachable	icmp	type 3		
ip all	ip	*		
ip forward	ipfwd	*		
ip mobile	ipmobile	*		
ip tunnel3	iptunnel	*		
ipv6 tunnel	iptunnel	IP protocol 41		
irc	tcp	port 6670		
	tcp	port 6680		
isakmp	udp	port 500		
kerberos	udp	port 88		
klm	rpc_udp	program no. 100020		
	pmap_udp	program no. 100020		
lpd	tcp	port 2766		
mountd	rpc_udp	program no. 100005		
	pmap_udp	program no. 100005		

TABLE C-1 SunScreen Services (Continued)

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
netbios datagram	udp_datagram	port 138		
netbios name	udp	port 137		
netstat	tcp	Port 15		
nfs acl	rpc_udp	program no. 100227		
	pmap_udp	program no. 100227		
nfs prog	pmap_udp	program no. 100003		
	tcp	port 2049		
	udp	port 2049		
nfs readonly prog	pmap_udp	program no. 100003		
	nfsro	port 2049		
nicname	tcp	port 43		
nlm	rpc_udp	program no. 100021		
	pmap_udp	program no. 100021		
			rpc_udp	program no. 100021
			pmap_udp	program no. 100021
nntp	tcp	port 119		
ntp	udp	port 123		
ntp-tcp	tco	port 123		
ospf	ip	type 89 (BROADCAST)		
pcnfsd	pmap_tcp	program no. 150001		
	pmap_udp	program no. 150001		
	rpc_tcp	program no. 150001		
	rpc_udp	program no. 150001		
ping	ping	port 8		
pmap tcp all	pmap_tcp	*		
pmap udp all	pmap_udp	*(BROADCAST)		
pop	tcp	ports 109–110		
printer	tcp	port 515		

TABLE C-1 SunScreen Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
quote	tcp	port 17		
radius	udp	port 1645		
real audio	realaudio	port 7070		
remote administration	tcp	ports 3852-3853		
rex	rpc_udp	program no. 100017		
	pmap_udp	program no. 100017		
rip	udp_datagram	port 520 port 520 (BROADCAST)		
rlogin	tcp	port 513		
	tcp_keepalive	port 513		
router announcement	icmp	type 9 type 9 (BROADCAST)		
router discovery	icmp	type 10		
		type 10 (BROADCAST)	icmp	type 9 type 9 (BROADCAST)
router solicitation	icmp	type 10 type 10 (BROADCAST)		
rpc all	rpc_udp	*		
rpc tcp all	rpc_tcp	*		
rquota	rpc_udp	program no. 100011		
	pmap_udp	program no. 100011		
rsh	rsh	port 514		
rstat	rpc_udp	program no. 100001		
	pmap_udp	program no. 100001		
rusers	rpc_udp	program no. 100002		
	pmap_udp	program no. 100002		
securid	udp	port 5500		
SecurID PIN	tcp	port 3855		
securidprop	tcp	port 5510		

TABLE C-1 SunScreen Services (Continued)

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
skip	iptunnel	type 57 type 79 * (BROADCAST)		
smtp	tcp	port 25		
snmp	tcp	port 161		
	udp	port 161		
snmp traps	udp_datagram	port 162		
spray	rpc_udp	program no. 100012		
	pmap_udp	program no. 100012		
sqlnet	sqlnet	port 1521		
ssh	tcp_keepalive	port 22		
ssl	tcp	port 443		
status	rpc_udp	program no. 100024		
	pmap_udp	program no. 100024		
StreamWorks	udp_datagram	port 1558		
			udp_datagram	port 1558
syslog	udp_datagram	port 514		
syslog	udp_datagram	port 514		
systat	tcp	port 11		
tcp all	tcpall	ports 0-3850 ports 3854-65535		
tcp-high-ports	tcp	ports 1024-65535		
telnet	tcp	port 23		
	tcp_keepalive	port 23		
tftp	udp	port 69 parameters (60 -1 7)		
time	tcp	port 37		
time-udp	udp	port 37		

TABLE C-1 SunScreen Services (Continued)

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
traceroute	udp_datagram	ports 33430–34000		
			icmp	type 11
			icmp	type 3
tsolpeerinfo_tcp	pmap_tcp	port 110002		
	rpc_tcp	port 110002		
tsolpeerinfo_udp	pmap_udp	port 110002		
	rpc_udp	port 110002		
udp_all	udpall	*		
udp-high-ports	udp	ports 1024–65535		
uucp	tcp	port 540		
VDOLive	tcp tcp	port 7000 port 7010		
			udp	port 32649
Vosaic	tcp	port 1235		
			udp_datagram	ports 61801–61820
			udp_datagram	ports 20000–20020
wais	tcp	port 210		
wall	rpc_udp	program no. 100008		
	pmap_udp	program no. 100008		
who	udp_datagram	port 513 (BROADCAST)		
whois	tcp	port 43		
www	tcp	port 80		
X11	tcp	ports 6000–6063		
	tcp_keepalive	ports 6000–6063		
ypbind	rpc_udp	program no. 100007		
	pmap_udp	program no. 100007		
yppasswd	rpc_udp	program no. 100009		
	pmap_udp	program no. 100009		
ypserv	nis	port 100004		

TABLE C-1 SunScreen Services (Continued)

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
	pmap_nis	program no. 100004		
	pmap_nis	program no. 100004 (BROADCAST)		
ypupdate	rpc_udp	program no. 100028		
	pmap_udp	program no. 100028		
ypxfrd	pmap_tcp	program no. 100069		
	pmap_udp	program no. 100069		
	rpc_tcp	program no. 100069		
	rpc_udp	program no. 100069		

* Service

The * service is a special type of *internal* service which has some of the characteristics of a service group. It includes a number of services, as shown in the list below, but those services are not displayed when you list services in the configuration editor or the GUI, and you cannot edit the services in *.

The * service, which acts as if each of its services were in separate rules, is designed to allow anything through, but it attempts to use the *best* service first, thereby providing better security. For example, the *ftp* state engine enforces the proper use of the stateful FTP protocols, in contrast to *ipmobile*, which does not inspect packets according to any of the stateful protocols. Note that *ipmobile*, which allows any IP traffic initiated by the source address, is the last service in the list of * services:

- nis
- pmap_nis
- pmap_dup
- pmap_tcp
- rpc_tpc
- rpc_udp
- realaudio
- rsh
- ftp
- tcp
- tcpall
- dns
- udp_datagram
- udp
- udpall

- ping
- icmp
- ipmobile

ah Service

IPsec Authentication Header (ah) uses IP protocol 51 and is used for traffic that has been authenticated using IPsec.

archie Service

SunScreen contains a service definition to handle the Archie UDP protocol. To screen Archie traffic, use the `archie` service.

CoolTalk Service

The CoolTalk service definition allows calls to be initiated but does not allow calls to be received. To receive calls, define a second rule with the addresses reversed. For example:

```
CoolTalk joe sam allow
CoolTalk sam joe allow
```

dns Service

DNS traffic consists of both UDP and TCP traffic. SunScreen includes a state engine to handle the UDP DNS protocol. TCP DNS is handled through the normal TCP state engine. To screen DNS traffic, use the predefined `dns` service.

esp Service

IPsec Encapsulating Security Payload (esp) uses IP protocol 50 and is used for traffic that has been encrypted or authenticated using IPsec.

ftp Service

The File Transfer Protocol (FTP) is used to copy files from one system to another. FTP is designed to work between hosts using different file structures and character sets.

SunScreen contains an `ftp` state engine to screen the FTP data connection. You specify the number for the FTP control port; the number for the FTP data port is one less than the FTP control port number. The predefined FTP service definition, `ftp`, uses the standard FTP control port number (21) and data connection port number (20).

FTP control connections time out after a period of inactivity. The FTP server typically closes the connect before this inactivity timeout occurs; however, if the timeout period elapses, the `quit` command can take 60 seconds or more to complete. During this time, FTP packets may be logged.

The `ftp` service supports both PASV and standard FTP connections. By default, the `ftp` service verifies that the FTP data port is 20 for standard FTP connections. To communicate with FTP servers that do not use port 20 for the data port, modify the `ftp` service definition to set its three parameters to: 600 600 1. The first parameter is the control session timeout (600 seconds). The second parameter is the data session timeout (600 seconds). The third parameter is a flag; a value of 1 specifies that the system will not verify that the FTP data port is 20.

Note that this does not affect PASV FTP sessions, because they never use port 20 for the data connection.

ICMP Packets

SunScreen provides predefined services for screening ICMP packets including `ping`.

The `icmp` state engine can also be used to create other services to pass ICMP messages of a specific type. Most of the common ICMP packets have entries in the predefined services.

These rules allow Inside systems to ping Outside systems, but not vice versa. It also allows ICMP unreachable packets to be sent from Outside systems to Inside systems. Note that the `ping` service allows packets in two directions (`ping-request` packets from Source to Destination and `ping-response` packets from Destination to Source) while the `icmp-unreach` service only allows packets to flow in one direction (from Source to Destination).

icmp Service

SunScreen includes predefined services for screening ICMP packets such as ping. These services use the icmp state engine and allow ICMP ping request-and-response exchanges between a Source and Destination system. Use the predefined service ping if you want to provide ping access.

You can use the icmp state engine to create other services to pass ICMP messages of a specific type. Most of the common ICMP packets have entries in the predefined services, as shown in the table:

Service	Source	Destination	Action
ping	Inside	Outside	allow
icmp-unreach	Outside	Inside	allow

These rules allow Inside systems to ping Outside systems, but block Outside systems from sending ping messages to Inside systems. It also allows ICMP unreachable packets to be sent from Outside systems to Inside systems. Note that the ping service allows packets in two directions (ping-request packets from Source to Destination and ping-response packets from Destination to Source), while the icmp-unreach service only allows packets to flow in one direction (from Source to Destination).

IP Packets

SunScreen can filter IP packets by IP protocol type alone. This is useful in special situations such as passing non-TCP/UDP protocols or when data are being encrypted.

To pass IP packets by protocol type, you need to define a new service using either the ip, ip tunnel, ip mobile, or ip fwd state engine. Specify the protocol of the packets you wish to pass. Note that protocol is always specified in decimal notation. If you specify * for the protocol, this means to pass *all* IP packets regardless of protocol type.

There are several predefined services included, such as skip (IP protocols 79 and 57), ip tunnel, ip mobile, and ip fwd.



Caution – Using one of the state engines with a protocol specification of * (any protocol), can be dangerous, because any traffic would be allowable. State engines should only be used in special cases or if the data are part of an encrypted tunnel.

The predefined IP services do not pass broadcast traffic. To pass broadcast traffic, you must define a new service or add broadcast to the predefined service.

ip Services

The `ip all` service is provided for backward compatibility with previous SunScreen products. You can achieve better performance by using either the `ip forward` (for IP traffic in one direction) or the `ip tunnel` (for IP traffic in both directions) services instead.

Example of the old method using `ip all`:

```
"ip all" host1 host2 allow
"ip all" host2 host1 allow
```

Example of the new method using `ip tunnel`:

```
"ip tunnel" host1 host2 allow
```

The `ip mobile` service is provided for use with mobile, remote clients. Like the `ip tunnel` service, `ip mobile` passes all IP traffic between a pair of addresses. Unlike the `ip tunnel` service, however, a rule specifying `ip mobile` forces the first connection to be made from the mobile client (a system with one of the addresses in Source Address).

Generally, `ip mobile` is used for SKIP-encrypted connections with the SKIP identity providing the authentication and access control. For example:

```
"ip mobile" Internet Mailhost SKIP-VERSION2
```

SunScreen can filter IP packets by IP protocol type alone. This is useful in special situations such as passing non-TCP/UDP protocols or when data are being encrypted.

If you want a Screen to pass IP packets by protocol type, you define a new service using either the `ip`, `ip tunnel`, `ip mobile`, or `ip fwd` state engine. Specify the protocol of the packets you wish to pass in decimal notation. If you specify `*` for the protocol, the service will pass *all* IP packets regardless of protocol type.

There are several predefined services included, such as `skip` (IP protocols 79 and 57), `ip tunnel`, `ip mobile`, and `ip fwd`.



Caution – Using one of the state engines with a protocol specification of `*` (any protocol), can be dangerous, because any traffic would be allowable. State engines should only be used in special cases or if the data are part of an encrypted tunnel.

The predefined IP services do not pass broadcast traffic. To pass broadcast traffic, you must define a new service or add broadcast to the predefined service.

ipsec Service

`ipsec` is a service group that comprises the three packet types that are used in IPsec secure communication.

isakmp Service

Internet Security Association and Key Management Protocol provides communication between security processes such as IKE key negotiation.

ipv6 tunnel Service

`ipv6` uses IP protocol 41 and carries encapsulated IPv6 packets over an IPv4 link such as the Internet.

nfs readonly Service

The `nfs readonly` service allows read-only access to the NFSv3.0 file system. Read-related functions, such as `lookup`, `read`, and `access`, are allowed. Functions that are not read-related, such as `rename` and `write`, are blocked; traffic is not permitted to pass under the `nfs readonly` rule.

ntp Service

To screen NTP traffic, use the `ntp` service. SunScreen contains a state engine to handle the NTP protocol. The source and destination UDP ports numbers are fixed at port 123. Broadcast NTP is not supported.

realaudio Service

SunScreen contains a service definition to handle RealAudio sessions. To screen RealAudio traffic, use the `realaudio` service.

rip Service

The Routing Information Protocol (RIP) is a dynamic routing protocol commonly used by Internet routers. RIP messages are carried in UDP datagrams. SunScreen includes a predefined service (`rip`) for passing RIP packets using the `udp-datagram` state engine with broadcast enabled. This means that a rule allows RIP packets (including broadcasts) from source to destination.

Enabling RIP in the default rule that passes RIP from the routers to all other addresses is usually sufficient. This enables the Screen to send and receive RIP packets without restriction. To restrict RIP traffic, do not enable RIP using the default access rules. Instead, define rules for RIP based on your security policy, for example:

Service	Source	Destination	Action
route	<i>routers</i>	*	allow
route	*	<i>routers</i>	allow

rpc Service

SunScreen contains a state engine to handle the RPC protocols. This can safely screen RPC protocol as long as they use the `portmapper` and do not use dynamic RPC program values.

To define a new RPC service, add a new service entry using both the `rpc_udp` and `pmap_udp` state engines. You specify the well-known RPC program of the RPC service you wish to pass. If you specify `*` for the RPC program, the service entry passes *all* RPC services, regardless of program.

Several well-known RPC services such as NFS and NIS have been defined to include all the RPC and non-RPC protocols that these systems require.

Some NFS clients use the lock manager. Because the lock manager makes connections in both directions (to NFS server and from NFS server), you may need to use the `nlm` service when you allow NFS access as shown in the following example:

Service	Source	Destination	Action
nfs	Inside	DMZ	allow
nlm	DMZ	Inside	allow

Broadcast port mapping (NIS) is not supported for encrypted connections.

smtp (Electronic Mail) Service

Simple Mail Transfer Protocol (SMTP) is used to send electronic mail between two message transfer agents using TCP. SunScreen includes a predefined service definition, `smtp`, to send and receive SMTP mail on TCP port 25.

sqlnet Service

SunScreen contains an `sqlnet` state engine to screen Oracle SQL*Net protocol. SQL*Net is Oracle's remote data access protocol that enables client-server and server-server communications across networks.

An Oracle client connects to the server using the port address of the listener, which is normally defined as TCP port 1521 during Oracle installation. `sqlnet` service is defined as using TCP port 1521. If Oracle is installed using a different port for the listener, you can modify the service definition for `sqlnet` service accordingly.

SQL*Net connections are established in two ways. An Oracle client connects to the listener using TCP port 1521, and the connection is established with the listener process. With Oracle multithreaded servers and prespawnd server processes, the client connects to the listener on TCP port 1521. The listener issues a redirect message back to the client containing an IP address and port number, and the client connects to this redirected IP address and port.

SunScreen supports both types of SQL*Net connections.

TCP Services

SunScreen screens TCP services by destination port numbers. Most common TCP services are already defined in the service entries supplied with SunScreen.

To define a new TCP service, define a new service entry specifying the `tcp` filter state system. Specify the destination TCP port or ports of the service you wish to pass. If you specify `*` for the port, the service will pass *all* TCP services regardless of port. Note that some services, such as FTP and RSH, cannot be passed in this way. They are not simple TCP protocols. They make additional connections in the reverse direction. These services must be specified as separate services if you wish to pass them.

The `tcp` state engine times out unused and silent connections five hours after a connection has been established. Some systems repeatedly retransmit until they receive an error about a terminated TCP connection. To send an ICMP rejection message, therefore, configure a rule using the `tcp` service, especially on your internal interfaces.

For example, the following rule allows `telnet` connections to be made from Inside systems to Outside systems.

Service	Source	Destination	Action
telnet	Inside	Outside	allow

traceroute Service

The `traceroute` service entry assumes that the UDP ports being used for `traceroute` are in the range of 33430-34000. If implementations of `traceroute` at your site use other ports, modify the port range as appropriate.

tsolpeerinfo Service

When two Trusted Solaris systems communicate with each other using the TSOL protocol, they typically use `rpc` program 110002 to exchange process attributes for peer processes. The entry in `/etc/rpc` is `tsolpeerinfo 110002 rpc.getpeerinfo peerinfod`.

If this service is blocked, services do not work. A connection is established, but Trusted Solaris waits for a response from `peerinfod` for additional information. Until it gets that response, the connection cannot proceed. The `tsolpeerinfo` service prevents this problem by ensuring that this service can be initiated from both sides of a connection through a firewall.

A server (`ftpd`, `telnetd`, etc., for example) spawned by `inetd` requests the audit attributes of a connecting client from a Trusted Solaris system. The server sends a `getpeerinfo` RPC back to the client, which responds with the required information. For example, to allow `telnet` through the firewall from HostA to HostB, but not from HostB to HostA, your rule base must include the following three rules:

- 1 `telnet HostA HostB ALLOW`
- 2 `tsolpeerinfo HostA HostB ALLOW`
- 3 `tsolpeerinfo HostB HostA ALLOW`

Without the `tsolpeerinfo` rules, the `telnet` connection appears to connect and hang. Note that if your rules involve encryption, the `tsolpeerinfo` rules must be modified to include the relevant encryption parameters as well.

Alternatively, you could define a group—`HostA+B`—containing both hosts. Rules 2 and 3 could then be combined to form the following rule:

- 2 `tsolpeerinfo HostA+B HostA+B ALLOW`



Caution – The `tsolpeerinfo` service does not work with dynamic NAT. Assume a client goes through a firewall and its address is dynamically changed with NAT. The server tries to `getpeerinfo` to the NAT address. Since this is a new connection initiated from a server that is unassociated with any state engine, this connection is dropped. There is no way to “de-NAT” the connection.

See the Trusted Solaris installation instructions in *SunScreen 3.2 Installation Guide* for details about installing SunScreen on a system running Trusted Solaris.

UDP Services

SunScreen contains several state engines to handle UDP protocols:

- `udp` – Provides stateful UDP packet filtering. Allows a single request-and-response exchange between source and destination. State entries time out in 20 seconds if no response is received.
- `udpall` – Identical to `udp`. It is useful for avoiding conflicts while defining service groups containing many services.
- `udp_datagram` – Passes UDP packets from source to destination. You can specify that broadcast packets should be passed.
- `udp_stateless` – Allows UDP packets to be sent between source and destination. The UDP Port(s) field specifies the list of destination UDP ports that are allowed. The source UDP port must be a unreserved port. Note that this is a two-way exchange of UDP packets.



Caution – Because some services use unreserved port numbers, use of this state engine can open up security holes. Its use is not recommended.

For all UDP engines, you define a new service entry specifying the well-known destination, UDP port. Specifying port `*` passes *all* UDP traffic.

VDOLive Service

The VDOLive service definition requires that the VDOLive clients be set to use a fixed port, which is port 32649 by default. You can modify the service definitions so that VDOLive will use another port.

www (World-Wide-Web Access) Service

The World Wide Web provides a graphical user interface that enables users to browse a global network of services and documents. SunScreen contains a predefined service definition for WWW that passes TCP connections on port 80.

Not all WWW services on the Internet use port 80; many reside on ports with other numbers, such as 8000 or 8080. If you only allow outbound WWW access under the `www` service entry, users cannot connect to all WWW resources. To compensate, you can define a new TCP service that enumerates additional nonstandard WWW ports you want to allow, or you can allow TCP access to all ports outbound using the default service.



Caution – Do *not* use the `tcp all` service to enable inbound `www` access to your public Web servers. This opens up a large security hole and allows outside users access to any TCP service on your systems. Instead, use a more restrictive service rule, such as the `www` service definition, with the port your Web server uses (generally port 80).

Network Service Groups

Network services can be organized into service groups, so that a single rule can apply to multiple network services. The table below lists the predefined service groups in SunScreen and the services that each group includes. Note that some services are members of more than one group, and other services are not included in any service group.

The `common` group compiles to list every service within the group in a specific order based on state-engine precedence. When a packet comes through, it tries to match each state engine in order of its precedence.

Note – See “* Service” on page 282 for information about the * service, which has some of the characteristics of a service group.

TABLE C-2 SunScreen Network Service Groups

Service Group Name	Member Services
common	tcp all
	udp all
	syslog
	dns
	rpc all
	nfs prog
	icmp all
	rip
	ftp
	rsh
	real audio
	pmap udp all
	pmap tcp all
	rpc tcp all
	nis
	archie
	traceroute
	ping
daytime	daytime
	daytime-udp
discard	discard
	discard-udp
echo	echo
	echo-udp
HA	HA heartbeat
	HA administration

TABLE C-2 SunScreen Network Service Groups *(Continued)*

Service Group Name	Member Services
ipsec	esp
	ah
	isakmp
mosaic	www
	ssl
	gopher
	ftp
netbios	archie
	netbios name
	netbios datagram
	netbios session
nfs	mountd
	nfs prog
	rquota
	nlm
	status
	nfs acl
nfs readonly	mountd
	nfs readonly prog
	rquota
	nlm
	status
	nfs acl
nis	ypserv
	yppasswd
	ypupdate
	ypbind
time	time
	time-udp

TABLE C-2 SunScreen Network Service Groups (Continued)

Service Group Name	Member Services
tsolpeerinfo	tsolpeerinfo_tcp tsolpeerinfo_udp

State Engines

SunScreen includes a number of state engines that act as protocol verifiers for services. For example, the `ftp` state engine checks port numbers when the `ftp` service is being used.

You cannot define new state engines, and you should not change which state engine is used by a predefined service. However, if you define a new service, you must specify the state engine the newly defined service will use.

Characteristics of State Engines

State engines have the following characteristics:

- **Connection management** – Each state engine understands the connection management of a particular protocol or set of protocols. State engines can be general, such as the `tcp` state engine (which allows a simple TCP connection) or specific, such as the `ftp` state engine (which understands the FTP protocol and parses FTP `PORT` and `PASV` commands).
- **Precedence level** – Each state engine has a precedence level. A service with multiple state engines (that is, a service group) is internally ordered by state engine. This order is given by the order in the `ss_stateengine` default list.
- **Discriminator value** – Each state engine has a *discriminator* value. This value is used to bind the state engine to a particular service. Examples are a port number for TCP and UDP services, an RPC program number for RPC services, or an `icmp` type for ICMP services.
- **Parameters** – Each state engine has a set of *parameters*. These parameters have a default value that can be overridden when the service is defined to modify the behavior of the state engine.

dns State Engine

The `dns` state engine is used for UDP DNS sessions. It looks inside the DNS responses and verifies that they have the same DNS ID as the request. The predefined service `dns` uses this state engine and is normally the only service to use it. Because the DNS service also uses the TCP protocol, the predefined service `dns` also has a second entry using the `tcp` state engine.

The discriminator for the `dns` state engine is the UDP port number of the DNS service. This is normally 53.

The `dns` state engine has two parameters:

- Response timeout in seconds – Specifies the time to wait for DNS responses. Default is 60 seconds. This can be changed (to 120 seconds, for example) in the configuration editor as follows:

```
edit> add service "dns" SINGLE FORWARD "tcp" PORT 53 FORWARD "dns" PORT 53 PARAMETERS 120
```

- Packet count – the number of response packets the state engine will allow for each outgoing packet. The default is 1; it is recommended that this not be changed.

ether State Engine

SunScreen in stealth mode can pass non-IP Ethernet frames between its interfaces. It cannot filter the frames on their content, but can pass (or drop) frames based on the frame "type." It can also determine which interfaces the frames are allowed to and from.

To pass non-IP traffic you need to define a new service entry using the `ether` state engine, specifying the type of protocol you wish to pass. The discriminator for the `ether` state engine is the frame type number in decimal.

The location of the value that you specify in the type field within the Ethernet packet depends on the Ethernet *frame type*. The following four Novell frame type designations, described below, are in common use:

- Ethernet II — Common name: Ethernet
- Ethernet 802.3 — Common name: "Raw" 802.3
- Ethernet 802.2 — Common name: 802.3
- Ethernet SNAP — Common name: 802.3/SNAP or 802.3/802.2/SNAP

Ethernet II — Common name: Ethernet

Ethernet II is the most common frame type and is used for TCP/IP as well as many other protocols. Ethernet type 0x8137 is used by IPX.

Destination Address	SourceAddress	Ethernet Type	Network Protocol Packet
6 bytes 0–5	6 bytes 6–11	2 bytes 12–13	up to 1500 bytes 14–1513

Ethernet 802.3 — Common name: “Raw” 802.3

Ethernet 802.3 has no protocol ID and can only carry IPX packets. It is distinguishable from Ethernet_802.2 only because the first 2 bytes of all IPX packets carried on Ethernet 802.3 must be all ones, which makes no sense in Ethernet 802.2. “Raw” 802.3 was the default frame type for NetWare software until NetWare v4.0 was released.

Destination Address	Source Address	Length	IPX Packet
6 bytes 0–5	6 bytes 6–11	2 bytes 12–13	up to 1500 bytes 14–1513; 0xFF 0xFF are the first two bytes

Ethernet 802.2 — Common name: 802.3

Note that the 802.2 header is implied by the 802.3 standard. Ethernet 802.2 is also known as: 802.3/802.2, to distinguish it from "raw" 802.3. It is used for OSI packets on 802.3 networks. Ethernet 802.2 is the default frame type for the NetWare v4.0 release. Values in parentheses in the table below are the values used by IPX.

Destination Address	Source Address	Length	DSAP (E0)	SSAP (E0)	Control (03)	Network Packet
6 bytes 0–5	6 bytes 6–11	2 bytes 12–13	1 byte 14	1 byte 15	1 byte 16	up to 1497 bytes 17–1513

Ethernet SNAP — Common name: 802.3/SNAP or 802.3/802.2/SNAP

Ethernet SNAP is an extension to 802.2, indicated by SAP value of hex AA. Ethernet SNAP is used by AppleTalk and is almost never used for IPX. Values in parentheses in the table below are the values used by IPX.

Dest. Addr.	Source Addr.	Length	DSAP 0xAA	SSAP 0xAA	Control 0x03	SNAP Header (0,0,0,81,37)	Network Packet
6 bytes 0-5	6 bytes 6-11	2 bytes 12-13	1 byte 14	1 byte 15	1 byte 16	5 bytes 17-21	up to 1492 b 22-1513

How SunScreen Checks the type Field

SunScreen checks the type field as follows:

- For Ethernet II packets, the type field specifies the value of the Ethernet type field located as offset 12 from the beginning of the packet. Any packet that has its Ethernet type field set to a value greater than 1526 is considered an Ethernet II packet. The range of applicable values for type is 1527 through 65536.
- For other Ethernet packets, the values of the DSAP and SSAP are examined, located at offsets 14 and 15 from the beginning of the packet. If the DSAP and SSAP are both 0xAA, the packet is assumed to be an Ethernet SNAP packet. For SNAP packets, the type field specifies the value of Ethernet type field located in the SNAP header at offset 20 from the beginning of the packet. The range of type values is 0 through 65536.
- If the DSAP and SSAP are not 0xAA, the type field specifies the value of the DSAP field, located at offset 14. The range of type values is 0 to 169 and 171 to 255; 170 (0xAA) is not allowed.

Example: Passing IPX Packets Between Host A and Host C

Imagine you want to pass IPX packets between HOST A and HOST C in the figure below:

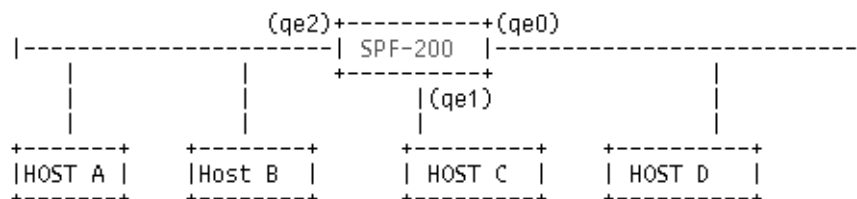


FIGURE C-1 Ether State Engine: Passing IPX Packets [NEW GFX NEEDED]

You have decided that the frame types used by these systems are 33079 & 33080 (hex 0x8137 and 0x8138).

1. Create and save new services using the ether state engine for each of these frame types. Create a service group (call it "ipx," for example) containing both of these services.

Note – The ether state engine takes a decimal value for type.

2. Pick an IP host on the qe2 interface and an IP host on the qe1 interface and create an address list called "qe1andqe2."

If you have defined interface objects for qe1 and qe2 (which you should do for anti-spoofing) these could be combined into a list called "qe1andqe2."

3. Define a rule:

Service: ipx
Source: qe1andqe2
Destination: qe1andqe2
Action: normal

This rule passes all frames with the specified types between the qe1 and qe2 interfaces. That is, a frame from any host on the network attached to qe2 (Host B, for example) will get passed to the network attached to qe1, if the type matches.

Note that there is no logging with the ether state engine, even if LOG_DETAIL is in the rule—because all SunScreen logging starts at the IP layer and there is no IP layer here.

ftp State Engine

The ftp state engine is used for FTP sessions. This state engine understands the control protocol used by FTP sessions including parsing PORT commands. It supports both traditional and PASV modes. The ftp service is typically the only service that uses this state engine.

The discriminator for the ftp state engine is the port number of the control connection, which is normally 21. The port number of the data session is always one less than the control connection unless this is overridden by the parameters below.

The ftp state engine has the following parameters:

- Lifetime of idle control session in seconds – Specifies the lifetime of an idle control session (default = 600 seconds)
- Lifetime of idle data session in seconds – Specifies the lifetime of an idle data session (default = 600 seconds)
- Flag value – Flag value is a set of bits. If bit 0x01 is set, non PASV data sessions are allowed to originate from a port other than one less than the control port. This feature is sometimes needed to communicate with FTP servers that incorrectly implement the FTP protocol so they do not need to run the data connection as root (default = 0).

icmp State Engine

The icmp state engine is used for ICMP protocols. It allows one-direction ICMP traffic to flow.

The discriminator for the icmp state engine is the ICMP type of the packet.

The icmp state engine has no parameters.

ip State Engine

The ip state engine is a stateless filter that passes unidirectional IP traffic of a particular IP type. The data can only flow in the forward direction (Source to Destination address) This state engine is supplied to provide backwards compatibility with the ip state engine in the SunScreen SPF-100. New service definitions should use either the ipfwd, iptunnel, or ipmobile state engines.

The discriminator for the ip state engine is the IP packet type.

The ip state engine has no parameters.

ipfwd State Engine

The ipfwd state engine allows unidirectional IP traffic of a certain IP type. The data can only flow in the forward direction (Source to Destination address)

The discriminator for the ipfwd state engine is the IP packet type.

The ipfwd state engine has the following parameters:

- Cache timeout in seconds – Specifies the amount of time before the system forgets about IP traffic between a pair of hosts (default is 60 seconds)
- Flag value – Must be 1

ipmobile State Engine

The ipmobile state engine allows bidirectional IP traffic of a certain IP type. The first connection must be initiated by the From address in the rule. Subsequent connections can be initiated from either side as long as the cache entry has not timed out.

The discriminator for the ipmobile state engine is the IP packet type.

The ipmobile state engine has the following parameters:

- Cache timeout in seconds – Specifies the amount of time before the system forgets about IP traffic between a pair of hosts (default is 3600 seconds or 1 hour).
- Flag value – Must be 0 (zero)

iptunnel State Engine

The `iptunnel` state engine allows bidirectional IP traffic of a certain IP type. Either side of the connection can initiate connections.

The discriminator for the `iptunnel` state engine is the IP packet type.

The `iptunnel` state engine has the following parameters:

- Cache timeout in seconds – Specifies the amount of time before the system forgets about IP traffic between a pair of hosts (default is 60 seconds)
- Flag value – Must be 0 (zero)

nis State Engine

The `nis` state engine is used to define services that are NIS UDP sessions. The predefined service `ypserv` uses the `nis` state engine and is normally the only service definition that uses this state engine.

The discriminator for this state engine is the RPC program number of the service. Normally, this is always 100004, the RPC program number for NIS.

The `nis` state engine has the following parameters:

- Response timeout in seconds – Specifies the time to wait for NIS responses. -1 specifies the state engine will wait forever (default = 60 seconds).
- Number of expected responses per request – Typically 1, because an NIS server sends only a single response to a request (default = 1).
- Flag value – If this value is set to 2, then the system accepts NIS responses from a different port than the NIS request port. This case occurs when an NIS server is responding to name lookups that is mapping to DNS entries (default = 2).

ntp State Engine

SunScreen contains a state engine to handle the NTP protocol. The source and destination UDP ports numbers are fixed at port 123. To screen NTP traffic, use the `ntp` service. Broadcast NTP is not supported.

ping State Engine

The ping state engine is used for an ICMP ping exchange. It allows ping requests in the forward direction and ping responses in the reverse direction.

The discriminator of the ping state engine is the ICMP type of the request packet. This is normally set to 8 to match that of an ICMP echo request packet.

The ping state engine has one parameter:

- Response timeout in seconds – Specifies the amount of time to wait for ICMP echo responses (default is 10 seconds).

pmap_nis State Engine

The pmap_nis state engine is used for the portmap protocol used by NIS services. It monitors NIS portmap requests and responses and builds a table of host/port to NIS service mappings. The ypserv service is typically the only service definition that uses the pmap_nis state engine.

The discriminator for the pmap_nis state engine is the RPC program number of the service. This is always 100004, which is the RPC program number for NIS.

The pmap_nis state engine has the following parameters:

- Response timeout in seconds – Specifies the time to wait for NIS portmap responses (default = 60 seconds)
- Lifetime of NIS portmap mapping entries in seconds – -1 specifies an infinite lifetime. Because NIS clients cache portmap information indefinitely at boot time, this value is normally set to -1 (default = -1).

pmap_tcp State Engine

The pmap_tcp state engine is used for the TCP portmap protocol used by TCP RPC services. It monitors the TCP portmap requests and responses and builds a table of hosts and ports to RPC service mappings. Normally, a service definition for a TCP RPC service requires both a pmap_tcp and a rcp_tcp state engine entry. The discriminator for the pmap_tcp state engine is the RPC program number of the service.

The pmap_tcp state engine has the following parameters:

- Response timeout in seconds – Specifies the time to wait for portmap responses. (Default = 60 seconds)

- Lifetime of portmap mapping entries in seconds – -1 specifies an infinite lifetime (default = 3600 seconds).

pmap_udp State Engine

The `pmap_udp` state engine is used for the UDP portmap protocol used by UDP services. It monitors the UDP portmap requests and responses and builds a table of hosts and ports to RPC service mappings. Normally, a service definition for a UDP RPC service requires both a `pmap_udp` and a `rpc_udp` state engine entry. The discriminator for the `pmap_udp` state engine is the RPC program number of the service.

The `pmap_udp` state engine has the following parameters:

- Response timeout in seconds – This parameter specifies the time to wait for portmap responses (default = 60 seconds)
- Lifetime of portmap mapping entries in seconds – -1 specifies an infinite lifetime (default = 3600 seconds or 1 hour)

realaudio State Engine

The `realaudio` state engine is used for RealAudio sessions. This state engine understands the control protocol used by these sessions including enabling the UDP ports used for the audio traffic. The `realaudio` service is typically the only service that uses this state engine. The discriminator for the `realaudio` state engine is the port number of the TCP control connection, which is normally 7070.

The `realaudio` state engine has one parameter:

- Lifetime of an idle control session in seconds – Specifies the lifetime of an idle control session (default = 3600 seconds)

rpc_tcp State Engine

The `rpc_tcp` state engine is used for RPC protocols that use the TCP protocol. Normally, a service definition for such a protocol requires both an `rpc_tcp` and `pmap_tcp` state engine entry. The discriminator for the `rpc_tcp` state engine is the RPC program number for the service.

The `rpc_tcp` state engine has one parameter:

- Idle session lifetime in seconds – Specifies the lifetime of an idle TCP RPC session in seconds (default = 86400 seconds or 24 hours)

rpc_udp State Engine

The `rpc_udp` state engine is used for RPC protocols that use the UDP protocol. Normally, a service definition for such a protocol requires both an `rpc_udp` and `pmap_udp` state engine entry. The discriminator for the `rpc_udp` state engine is the RPC program number for the service.

The `rpc_udp` state engine has the following parameters:

- Response timeout in seconds – Specifies the time to wait for RPC responses. `-1` specifies to wait forever (default = 60 seconds)
- Number of expected responses per request (default is 1)
- Flag value – Specifies whether RPC responses must come from the same host or port that the RPC request specified. If the `0x01` bit is set, RPC responses from a different host than the request are allowed. If the `0x02` bit is set, RPC responses from a different port than the request port are allowed (default = 0)

rsh State Engine

The `rsh` state engine is used for remote shell (`rsh`) sessions. This state engine understands the control protocol used by these sessions, including the enabling of the TCP connection used for `stderr` messages. The `rsh` service is typically the only service that uses this state engine. The discriminator for the `rsh` state engine is the port number of the RSH server. This is normally 514.

The `rsh` state engine has one parameter:

- Lifetime of idle session in seconds – Specifies the lifetime of an idle session (default = 86400 seconds or 24 hours)

sqlnet State Engine

The `sqlnet` state engine is used for Oracle SQL*Net sessions.

It understands the network protocol used by SQL*Net, including redirected sessions (see “`sqlnet` Service” on page 289). The `sqlnet` service is typically the only service using the `sqlnet` state engine. Its discriminator is the port number of the Oracle listener, which is normally TCP port 1521.

The `sqlnet` service is typically the only service using this state engine.

tcp State Engine

The `tcp` state engine is used for TCP sessions. This state engine allows simple TCP connections. It cannot handle protocols such as FTP or RSH that have more complicated connection management protocols, especially if they open connections in the reverse direction. In those cases, the appropriate, more specific state engine should be used.

The discriminator for the `tcp` state engine is the port number of the TCP service.

The `tcp` state engine has one parameter:

- Lifetime of idle connection in seconds – Specifies the lifetime of an idle connection (default = 86400 seconds or 24 hours)

tcp_keepalive State Engine

The `tcp_keepalive` state engine is for use with protocols that spend long periods in an idle mode (`telnet`, for example). This state engine prevents the statetable entry from timing out if no packets are sent for a long time. Some SunScreen services (`telnet`, `rlogin`, `ssh`, `X11`) use `tcp_keepalive` by default. `tcp_keepalive` should be used for any TCP-based service that by its nature can include long periods of idle time.

`tcp_keepalive` causes the Screen to emit a “fake” keepalive packet to the session’s source host, claiming to have been sent by the session’s destination host. The keepalive packet is sent a few minutes before the Screen normally would drop the session. If the source host is still alive, it responds with an ACK, which causes the Screen to rejuvenate the session lifetime. The ACK is forwarded to the destination host, which responds if it is still alive. If either host has reset or timed out its end of the connection, it will respond with an RST, which causes the Screen to discard the session.

The `tcp_keepalive` state engine definition specifies that the first keepalive probe be sent 15 minutes before the session expires. If there is no response, multiple probes are sent, rapidly at first, then slowing: 900, 880, 860, 820, 740, 580, and 260 seconds before the session expires.



Caution – If you use this state engine for a service, it could lead to a connection being left open through the firewall for an extended period of time. Imagine, for example, someone telnets through the firewall, leaves the connection sitting at a prompt, and then goes on vacation for two weeks. Keepalive probes will continue to be successfully sent and the connection will stay open for two weeks.

It is up to the security administrator of the site to determine if use of this state engine is appropriate. Use of this state engine coupled with an inactivity timeout on login sessions would prevent such a situation from occurring and would make the firewall much more transparent to users, as there would be no "hung" sessions. Careful consideration should be given to the tradeoff between risk and convenience.

The `tcp_keepalive` state engine has two parameters:

- Lifetime of idle connection in seconds – Specifies the lifetime of an idle connection (default = 86400 seconds or 24 hours)
- Number of seconds before timeout that keepalive probes start

`tcpall` State Engine

The `tcpall` state engine is used for TCP service definitions that specify a large range of ports such as the predefined service `tcp all`. because it has a lower precedence than `tcp`, `ftp`, `rsh`, or `realaudio`, it does not override any of those services. Normally, this state engine is only used for the predefined service `tcp all`.

The discriminator for the `tcpall` state engine is the port number of the TCP service.

The `tcpall` state engine has the following parameter:

- Lifetime of idle connection in seconds – Specifies the lifetime of an idle connection (default = 86400 seconds or 24 hours)

`udp` State Engine

The `udp` state engine is used for UDP services. It allows one or more responses to a UDP request. The requests are validated to make sure they come from the correct address and port and are sent to the correct address and port. The response source address and port checking can be modified using the parameters below.

The discriminator for the `udp` state engine is the port number of the UDP service.

The `udp` state engine has the following parameters:

- Response timeout in seconds – Specifies the amount of time to wait for UDP responses. -1 specifies an infinite response timeout. (Default = 60 seconds).
- Number of responses per request – Specifies the number of expected responses for each request. If the number of response specified is 0, any number of responses can be received and the session terminates only after an idle period when no responses have been received. Never specify both a response time of -1 and 0 for the number of responses per request.
- Flag value – Specifies valid sources for UDP responses.
 - If bit 0x01 is set, the UDP response can come from a different host than the request, which is useful for UDP services on multihomed servers that respond using a different address.
 - If bit 0x02 is set, the UDP response can come from a different port than the request.
 - If both bit 0x02 and bit 0x04 are set, then requests can come from a different port than the request, and subsequent requests can also use that new port. This is useful for handling TFTP servers that sometimes switch ports in mid-session.

udpall State Engine

The `udpall` state engine is used for UDP services where a large number of ports are specified. It has a lower precedence than the `dns` and `udp` state engines and does not override services defined with those state engines. It allows one or more responses to a UDP request. The requests are validated to make sure they come from the correct address and port and are sent to the correct address and port. The response source address and port checking can be modified using the parameters below.

The discriminator for the `udpall` state engine is the port number of the UDP service.

The `udpall` state engine has the following parameters:

- Response timeout in seconds – Specifies the amount of time to wait for UDP responses. -1 specifies an infinite response timeout (default = 60 seconds).
- Number of responses per request – Specifies the number of expected responses for each request. If the number of response specified is 0, any number of responses can be received and the session terminates only after an idle period when no responses have been received. Never specify both a response time of -1 and 0 for the number of responses per request.
- Flag value – Specifies valid sources for UDP responses.
 - If bit 0x01 is set, the UDP response can come from a different host than the request, which is useful for UDP services on multihomed servers that respond using a different address.

- If bit 0x02 is set, the UDP response can come from a different port than the request.
- If both bit 0x02 and bit 0x04 are set, then requests can come from a different port than the request, and subsequent requests can also use that new port. This is useful for handling TFTP servers that sometimes switch ports in mid-session.

udp_datagram State Engine

The `udp_datagram` state engine is used for one-way UDP protocols. It allows UDP packets to pass in the forward direction only. It is used for services that send UDP packets in one direction, such as `syslog`.

The discriminator for the `udp_datagram` state engine is the port number of the UDP service.

The `udp_datagram` state engine has no parameters.

udp_stateless State Engine

The `udp_stateless` state engine is used for stateless UDP session filtering. This engine is included for backwards compatibility with older SunScreen products. It has been replaced in most cases with stateful UDP filtering. Because this engine is stateless UDP packet filtering, services defined using it cannot safely validate that the responses go to the same port as the request.

The discriminator for the `udp_stateless` state engine is the port number of the UDP service.

The `udp_stateless` state engine has no parameters.

Error Messages

This appendix describes the error messages generated by various components of the SunScreen software—`ssadm edit`, `ssadm activate`, and `ssadm lock`—and a suggested solution for each error condition. The following topics are included:

- “Error Messages From `ssadm edit`” on page 309
- “Error Messages From `ssadm activate`” on page 313
- “Error Messages From `ssadm lock`” on page 321

Error messages may be displayed while editing the address, rule, and service configurations (and from the corresponding GUIs).

The expression, *[ARGUMENTS]*, used in the following error messages means that the same set of arguments passed into `ssadm*` is echoed back.

For example, if you type: "add address a b junk x y z" the error message is: "add address a b junk x y z: error_message".

Error Messages From `ssadm edit`

- Usage: `ssadm edit [-beim] [-I ip] [-U user] policy [-c command]`

Return code: 1

You invoked the edit program incorrectly.

- Error illegal policy name *policyname*

Return code: 1

You specified an illegal policy name while invoking the editor.

- Discarding unsaved changes

Return code:

You chose to QUIT even though there are unsaved changes.

- Write Lock Held

Return code:

You asked for the lock_status, and you currently hold the write lock.

- Read Lock Held

Return code:

You asked for your lock_status, and you currently hold the read lock.

- Registry Version: #Policy Version: #

Return code: 0

You asked for the version currently being edited.

- no longer supported. use edit del request syntax

Return code: 251

You tried to delete a NAT Rule using the old `ssadm nat` or `ssadm access` command.

- There are unsaved changes

Return code: 231

You attempted to quit without saving unsaved changes.

- Cannot name Policy lock

Return code: 232

You attempted to save the policy as lock, which is a reserved word.

- There are unsaved changes Version.

Return code: 233

You attempted to save the policy as version, which is a reserved word.

- There are unsaved changes in Registry

Return code: 234

You attempted to save the policy as registry, which is a reserved word.

- Cannot save a versioned file with a new name

Return code: 235

You attempted to save a versioned policy with a new name.

- Registry objects redefined

Return code:

An entry in the registry (on disk) has more than one definition. All definitions after the first are lost upon the next save.

- file not found

Return code: 240

The policy to be read did not exist.

- parse error

Return code: 241

The policy or registry file on disk is corrupt and cannot be read. Make sure you have a backup or a recent version saved.

- could not acquire read lock

Return code: 242

The configuration editor could not acquire a read lock. Likely, the lock file is corrupt or some process is hanging. `ss_lock -c policy` is likely to be needed.

- could not acquire write lock

Return code: 243

A request to gain the write lock failed. Likely some other process currently holds the write lock.

- lock not held

Return code: 244

An attempt was made to save changes, but this process no longer holds the write lock. Perhaps someone else has issued an `ss_lock -c policy` and invalidated the lock.

- cannot modify "*"

Return code: 247

An attempt was made to modify the Address or Screen object *. This is a reserved name and cannot be modified.

- cannot modify "localhost"

Return code: 248

An attempt was made to modify the Address *localhost*. This is a reserved name and cannot be modified.

- lock unavailable

Return code: 249

Indicates that something happened to the lock files. *ss_lock -c policy* is likely needed..

- unresolved references

Return code: 250

A reference is made to a named object in the global registry that does not exist in the registry.

- invalid input

Return code: 251

A request was not well-formed.

- unknown operation

Return code: 252

A request was issued for an invalid data type.

- internal error

Return code: 253

A request used an invalid operation.

- Error: invalid input

Return code : nonzero :nonzero

- Warning: Adding ADMIN Interface to an routing machine.

You added an ADMIN interface to a routing machine. You probably want this to be a ROUTING interface.

- (ssadm interfaces) Warning: operation replaced only routing Interface

Appears as a result of an "add Interface" request

- (ssadm interfaces) Warning: operation replaced only stealth Interface

Appears as a result of an “add Interface” request

- (ssadm interfaces) Warning: operation left only one stealth Interface

Appears as a result of an “add Interface” request

Error Messages From ssadm activate

- Error output directory does not exist *output directory*

Return code : nonzero

You provided an invalid output directory name with ssadm activate.

- Too many Time objects being used. Limit is 31.

Return code: 236

The policy being compiled and activated refers to more than 31 distinct time objects.

- Registry objects redefined

Return code:

An entry in the registry (on disk) has more than one definition. All definitions after the first are lost upon the next save.

- Screen object not found

Return code: 239

The -S passed to ssadm activate is a nonexistent Screen object.

- file not found

Return code: 240

The policy to be read did not exist.

- parse error

Return code: 241

The policy or registry file on disk is corrupt and cannot be read. Be sure you have a backup or a recent version saved.

- compile error

Return code: 245

No longer used.

- unresolved references

Return code: 250

A reference is made to a named object in the global registry that does not exist in the registry.

- Error: Status NAT, but Addresses are different sizes (NAT entry)

Return code : nonzero

- Error: Original and Translated Source Intersect

Return code : nonzero

- Error: Original and Translated Destination Addresses Intersect

Return code : nonzero

- Error: Cannot translate both source and destination addresses

Return code : nonzero

- Screen object must define smtp address

Return code : nonzero

The Screen object must define the SMTP Address if the SMTP proxy is to be used.

- Error: Service not defined (remote administration)

Return code : nonzero

The indicated service is needed by the system, but the definition has either been deleted or renamed in the global registry.

- Error: SunScreen object has no Administrative Certificate *Screen name*

Return code : nonzero

The Screen object is not fully defined. Remote administration is indicated but the Screen is lacking a Certificate.

- Error: SKIP and Ethernet filtering not supported

Return code: nonzero

An Ethernet-based Rule is specified (that is, a service that includes the “ether” state engine) and it also indicates SKIP is to be used.

- Error: More than 16 Interfaces defined for a given type

Return code: nonzero

Only 16 of a given type of interface are supported.

- Could not find HA Service

Return code: nonzero

HA is indicated but the services “HA administration” and “HA heartbeat” have been either removed or renamed.

- Error: HA cluster missing HA IP Addresses

Return code: nonzero

The Screen objects participating in the current HA cluster lack HA_IP addresses.

- Error: HA IP address is not on HA interface

Return code: nonzero

The HA IP address specified is not part of the HA interface.

- Error: Service incorrectly defined

Return code: nonzero

A service has contradictory information, such as the same port but different state engines, or different parameters.

- Error: Interfaces intersect

Return code: nonzero

Two (or more) interfaces’ addresses intersect.

- Error: Rule uses Certificate Group with Service containing Reverse Filter *rule*

Return code: nonzero

The reverse rule swaps the certificates, and groups are not supported in the encrypting case.

- Error: Rule uses Certificate Group with Service containing Reverse State Engine *RULE*

Return code: nonzero

The reverse rule swaps the certificates, and groups are not supported in the encrypting case.

- Error: Service not defined (remote administration)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (snmp traps)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (skip)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (certificate discovery)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (rip)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (dns)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: Service not defined (nis)

Return code: nonzero

The service is needed internally but has been either renamed or deleted.

- Error: could not generate Rule from Screen: *Screen name*

Return code: nonzero

- Error: could not generate Rule to Screen: *Screen name*

Return code: nonzero

- Error: *Screen name* is missing encryption parameters

Return code: nonzero

- Error: *Screen name* requires a certificate to administer Screen
Screen name

Return code: nonzero

- Error: *Screen name* requires a certificate to be administered by
Screen *Screen name*

Return code: nonzero

- Error: HA Secondary with no Master

Return code: nonzero

HA is indicated, but no primary Screen is specified.

- Error: Incomplete Screen definition

Return code: nonzero

One of the following is missing given that HA_Secondary is indicated: certificate, key, data, mac, or compression algorithm.

- Error: HA enabled with no HA Interface defined

Return code: nonzero

- Error: HA enabled with multiple HA Interface(s) defined

Return code: nonzero

- Error: HA not enabled but HA Interface(s) defined

Return code: nonzero

- datacompiler: Error writing data file (fseek failed)

The data compiler could not write the output file owing to a failed fseek.

- Error: Remote Certificate *name1* uses multiple local
certificates: *name2* and *name3*

A certificate *name1* that is not local to this Screen is used in at least two SKIP_VERSION_1 rules, but the local certificate is not the same. SunScreen supports only using a one local certificate for any given remote certificate in SKIP_VERSION_1 compatibility mode. You must either use skip_version_2 or change one of *name2* and *name3* to the other.

- datacompiler: Manual Table too large. Limit is 65535.

There are more than 65535 pairs of certificates for either manual keying or support SKIP_V1 nodes. There is a limit of 65535.

- `datacompiler: Skip_V1 Table too large. Limit is 65535`

There are more than 65535 pairs of certificates for either Manual Keying or support SKIP_V1 nodes. There is a limit of 65535.

- `Activation failed, error error code`

Return code: 1

- `Active configuration: NAMEActivated by user on date`

Return code: 0

- `Warning: Rule type could not be determined and is being discarded Svc addr . . .`

A problem determining how to implement the rule occurred and is being discarded.

- `TYPE: name is already defined. Redefined on line <#> in file file`

TYPE is address, action, service, state engine. This means that the *name* is defined multiple times. One of the definitions must be removed. Using the `ssadm*` command removes the first such definition. To remove the second, and keep the first intact, you must use a text editor on the file on the Screen.

- `Error: name1 is not defined. Used on line # [in file file] by name2`

Indicates an unresolved reference, where *name2* refers to *name1* but *name1* is not defined. You need to define *name1* or remove the reference by modifying *name2*.

- `Address name is part of a cycle`

A circular reference in an address list definition, such that list A includes list B as a member and list B includes list A as a member. You must break the cycle for the compilation to be successful.

- `Service name is part of a cycle`

A circular reference in a service list definition, such that list A includes list B as a member and list B includes list A as a member. You must break the cycle for the compilation to be successful.

- `Service incorrectly defined. name...`

The service is internally inconsistent. Either the service defines two state engines in the same class and subclass for the same port, or the same port and the same state engine are used twice but with different parameters. You must redefine this service for the compilation to be successful.

- `Error: "name" is not defined. Used on line # in file FILE by name`

You are referring to an object (address, service) that has not yet been defined.

- Invalid Domain Name: "*name*"

You have entered a domain name that has illegal characters, such as /. Use the default domain name "default."

- Error: Domain "*name*" does not exist.

You have entered a nonexistent domain name. Use the default domain name "default."

- unknown operation

You have requested an operation that is not recognized.

- Sorry, *character* character not supported.

You have entered an unsupported character.

- could not acquire lock to read data, please reissue request.

Too many concurrent processes are running.

- could not acquire lock to write data, please reissue request.

Too many concurrent processes are running.

- invalid input

You entered something incorrectly. Refer to the relevant man page to verify you have the correct command syntax.

- unknown data type.

You requested an operation on an unknown data type.

- Error: "*name*" cannot be a Local Certificate.

The first certificate specified is supposed to be the Administration Station's certificate. If the certificate is local to the Screen, then it cannot be the Administration Station's certificate.

- Error: Missing Remote Certificate: "*name*"

The first certificate could not be found in the Certificate registry, as maintained by ssadm certificate. Be sure the entry is entered correctly.

- Error: "*name*" must be a Local Certificate.

The second certificate must belong to the Screen. Try again and verify that the second certificate is the Screen's certificate.

- Error: Could not find Local Certificate: "*name*"

The second certificate could not be found in the Certificate registry, as maintained by ssadm certificate. Be sure the entry is entered correctly.

- cannot modify Address *

You attempted to modify *, which is not user-editable.

- cannot modify Address "localhost"

You attempted to modify localhost, which is not user-editable.

- Error: Service "SERVICE" not found

The user-indicated service is missing.

- Warning: *RULE* uses invalid pair of certificate and will be ignored

A SKIP-based rule must include one local and one nonlocal certificate. If both are local, or both are nonlocal, then the rule is invalid and will be ignored. If you believe the rule is necessary for this Screen, verify that one of the certificates is local and one is nonlocal, and reactivate.

- Warning: *PROXY* proxy server not found. No rule generated

You specified proxy definition cannot be found and a proxy rule was specified. The rule necessary to support the proxy cannot be generated. Be sure the appropriate proxy server is defined.

- Error: Configuration "*name*" does not exist in domain *name*

The configuration does not exist.

- Error: # NAT entries are incorrectly defined

The NAT entry is invalid if its public and private addresses intersect with each other or any other address in the NAT table. Be sure that no two NAT entries intersect.

- Could not find HA Service

The service "HA Service" could not be found.

- Service incorrectly Defined: *SERVICE*

The specified service is not well defined. For example, it may specify the same port for multiple state engines that conflict, like UDP and UDP-datagram.

- Error: Interface does not exist (1e0)

1e0 is the name of the nonexistent interface. This happens if the global common registry being activated contains an interface that the machine that is doing the compile and activate does not have.

- Warning: Could not verify Interface "*name*" exists

You added an interface and it could not be verified on the Screen.

- Expecting "{" but found a *character*

The syntax entered was incorrect. See the man page for correct syntax.

- Expecting "}" but found a *character*

The syntax entered was incorrect. See the man page for correct syntax.

- Unexpected end of input

The syntax entered was incorrect. See the man page for correct syntax.

- Invalid Range specified # - #

You entered a range where the end value was less than the start value.

- Service definition is internally inconsistent

You specified service is not well-defined. For example, it may specify the same port for conflicting state engines, such as UDP and UDP-datagram.

Error Messages From ssadm lock

The ssadm lock component's error messages follow :

- Usage: ssadm lock -w | -c policy

Return code: 1

You invoked ssadm lock command incorrectly.

- lock held by *user @ IP* process id *pid*

The lock is held by this UNIX process with process ID *pid*.

Logged Packet Reasons

This appendix lists and describes codes and messages for logged packets.

Why Codes

The table below lists common reasons for logging packets in the SunScreen log and in the SNMP syslog files. A number below 256 indicates that the packet passed. A number of 256 or greater indicates that the packet was dropped. The reason numbers listed here are sometimes referred to as “why codes.”

TABLE E-1 Logged Packet Reasons

Number	Log Error Message	SNMP Error Message	Explanation
1	Passed packet logged	passed(1)	Packet passed. The packet was passed by a rule that specified the packet should also be logged.
256	Denied or no pass rule found	noRuleOrDenyRule(256)	Packet dropped because it did not match any rule. Can also indicate that the packet’s source address was invalid for the network interface.

TABLE E-1 Logged Packet Reasons *(Continued)*

Number	Log Error Message	SNMP Error Message	Explanation
257	No connection	noState(257)	Packet dropped due to missing state information. The packet was part of an existing, possibly legal session, but no session information could be found. This could be due to the Screen timing out the connection, the Screen being rebooted and losing session state, or a protocol violation where the initial packets were not sent.
258	Out of memory	noMemory(258)	Packet dropped due to the lack of Screen memory. The Screen could not create the session state due to a lack of real memory. The Screen will accept new sessions when current sessions are closed.
259	Too many connections	tooManySessions(259)	Packet dropped because the maximum number of sessions are already open. The Screen will accept a new session when a current session of this type is closed.
260	Invalid port	invalidPort(260)	Packet dropped due to an invalid port number specification. An example is an FTP data session not on port 20.
261	Bad format	invalidFormat(261)	Packet dropped due to an invalid format. The Screen determined that the packet did not match the service specified in the rules.
262	Bad direction	invalidDirection(262)	Packet dropped due to invalid "direction." For example, a DNS request was received when a DNS response was expected.
263	Too many responses	tooManyResponses(263)	Packet dropped due to too many responses. The applicable rule specified a simple UDP exchange but the Screen received multiple responses.
264	Too short	tooShort(264)	Packet dropped because it was too short for the service specified.

TABLE E-1 Logged Packet Reasons *(Continued)*

Number	Log Error Message	SNMP Error Message	Explanation
265	Bad protocol	invalidProtocol(265)	Packet dropped because of an invalid protocol identifier. For example, an RPC packet was not of protocol UDP or TCP.
266	No port map	noPortmapEntry(266)	RPC packet dropped due to lack of port mapping entry. An RPC packet was received on an invalid port. This can occur when the Screen times out RPC portmap entries faster than the end nodes.
267	Bad port map	invalidPortMapEntry(267)	RPC packet dropped due to invalid port mapping entry. The portmapper specified that a different RPC program resides on the port.
268	NIS protocol error	nisProtocolError(268)	NIS+ packet dropped due to protocol error (not implemented).

TABLE E-1 Logged Packet Reasons (Continued)

Number	Log Error Message	SNMP Error Message	Explanation
269	Bad interface	invalidInterface(269)	<p>Indicates a “bad policy.” This error message is typically caused by an invalid identity. The packet was dropped because the encryption characteristics of the packet did not match those specified in an otherwise matching rule. That is, the source address, destination address, and service of the packet matched at least one rule, but the encryption setting conflicted with what was received. Possible encryption characteristic differences include the following:</p> <ul style="list-style-type: none"> ■ The packet was received encrypted, but the rule specified that it must be unencrypted. ■ The packet was received unencrypted, but the rule specified that it must encrypted. ■ One of the encryption parameters of the packet did not match a parameter specified for the rule. For example, a mismatching key algorithm was used or the wrong certificate was specified. <p>The encryption settings for the sender and the Screen should be compared to verify that they are identical and that the correct keys are being used.</p>
270	Bad policy	invalidPolicy(270)	A SKIP packet matched an existing encryption rule but had one or more parameters set incorrectly.
272	Bad source address	invalidSourceAddres(272)	Indicates a packet was dropped because it was received on an interface where it was not expected; that is, the packet was dropped owing to spoof-detection checks. If the source of the rejected packet is supposed to be allowed on the interface, it should be added to the address group assigned to the interface.
274	Fragment too big	fragmentTooBig(274)	Indicates a possible network attack.

TABLE E-1 Logged Packet Reasons (Continued)

Number	Log Error Message	SNMP Error Message	Explanation
275	Fragment overlap	fragmentOverlap(275)	A packet was fragmented while it was in transit and the fragments contain redundant data. May indicate a network attack.
277	cert not in rule	certNotInRule(277)	An inbound packet was decrypted for which SKIP identities, algorithms, or version mismatched its rule in the active policy. The packet was dropped. (See also Number 269 above.)
278	attempt to encrypt a decrypted packet	invalidEncrypt(278)	An inbound packet was decrypted for a rule which only indicates encryption. The packet was dropped. (See also Number 269 above.)
279	no state associated with policy	noSKIPState(279)	An inbound packet was decrypted for which no rule or state exists in the active policy. The packet was dropped.
280	stale skip policy	staleSKIPPolicy(280)	An inbound packet was decrypted for an old (stale) state entry. The packet was dropped.
281	illegal dest address	invalidDestinationAddress(281)	An outbound packet was dropped because the destination was illegal on the interface of a screen with destination address checking enabled (DEST_CHECK).

Glossary

active Screen	Screen in a high availability cluster that is keeping state and passing traffic. There is always exactly one active Screen in a correctly operating high availability cluster. See primary Screen and passive Screen.
address	In networking, a unique code that identifies a node to the network. SunScreen uses IP addresses.
ACE/Server	The management component of securID.
ADP	Algorithm Discovery Protocol. Enables one entity to inform another of the capabilities it supports.
AH	Authentication Header. A mechanism for providing strong integrity and authentication for IP datagrams.
algorithm	Sequence of steps designed to solve a problem or execute a process such as drawing a curve from a set of control points, or encrypting a block of data.
AMI	Authentication Management Infrastructure.
API	application program interface. Set of calling conventions defining how a service is invoked through a software package. An interface between the operating system and application programs, which includes the way the application programs communicate with the operating system, and the services the operating system makes available to the programs.
argument	Item of information following a command. It may, for example, modify the command or identify a file to be affected. Sometimes the term parameter is used.
ATM	asynchronous transfer mode. Transmits data, voice, video, and frame relay traffic in real time. With ATM, digital information is broken up into standard-sized packets, each with the address of its final destination.

attack	Attempted cryptanalysis or an attempt to compromise system security.
authentication	Property of knowing that the claimed sender is in fact the actual sender.
broadcast	Packet delivery system, where a copy of a given packet is distributed to all hosts attached to the network.
CA	See <i>certificate authority</i> .
cache	Buffer of high-speed memory used to store frequently accessed memory or values. A cache increases effective memory transfer rates and processor speed.
CBC	Cipher Block Chaining (see also DES). A mode used to chain a feedback mechanism, which essentially means the previous block is used to modify the encryption of the next block.
CDP	Certificate Discovery Protocol. Request and response protocol used by two parties to transfer certificates.
Centralized Management group	Multiple secondary Screens that are managed by the Centralized Management group's primary Screen. Note that a Screen in a centrally managed group, whether primary or secondary, can also be part of a HA cluster. See HA cluster.
certificate	Data structure that binds the identity of an entity with a public-key value.
certificate authority	Trusted network entity that digitally signs a certificate containing information identifying the user; such as, user's name, issued certificate, and the certificate's expiration date.
certificate identifier (ID)	Generic naming scheme term used to identify a particular self-generated or issued certificate. It effectively decouples the identification of a key for purposes of key lookup and access control from issues of network topology, routing, and IP addresses.
CFB	Cipher Feedback. Uses a block cipher to implement a stream cipher.
cipher	Cryptographic algorithm used for encryption or decryption.
ciphertext	Encrypted message.
cluster	Screens in an HA cluster connected by a high-speed network that work together as if they were one Screen. See high availability.
common objects	Data objects that are relevant to all SunScreen policies. They include: address, screen, state engine, service, interface, certificate, time, and VPN gateway groups.
confidentiality	Property of communicating such that only the sender and the intended recipients know what is being sent, and unintended parties cannot determine what is sent.

configuration	Union of one policy with the common objects to form a complete description of the behavior of one or more Screens.
content filtering	Practice of allowing or disallowing traffic based on the content of the data being sent.
decryption	Process of converting ciphertext back to plaintext.
demilitarized zone	Small protected inside network or subnetwork that provides limited public access to resources such as web servers, FTP servers, and other information resources.
DES	Data encryption standard. A common algorithm for encrypting and decrypting data.
DMZ	See demilitarized zone.
DNS	domain naming system. Distributed name and address mechanism used in the Internet.
DST	Destination addresses.
dynamic packet filtering	See stateful packet filter.
dynamic translation	NAT converts a set of internal private addresses into external public addresses. It allows internal hosts to contact external hosts, but cannot be used to allow external hosts to contact internal hosts.
encapsulation	Technique used by layered protocols in which a layer adds header information to the protocol data unit from the layer above. In Internet terminology, for example, a packet would contain a header from the physical layer, followed by a header from the network layer (IP), followed by a header from the transport layer (TCP), followed by the application protocol data. See tunneling.
encryption	Process of protecting information from unauthorized use by making the information unintelligible. Encryption is based on a code, called a key, which is used to decrypt the information. Contrast with decryption.
ESP	Encapsulating Security Payload. Mechanism for providing integrity and confidentiality to IP datagrams. In some circumstances it can also provide authentication to IP datagrams, depending on which algorithm or algorithm mode is used. It does not provide nonrepudiation and protection from traffic analysis.
Ethernet	LAN that enables real-time communication between machines connected directly through cables.
failover	Process by which a passive Screen in a high availability group becomes the active Screen if the active Screen becomes unavailable.
filter	Program that reads the standard input, acts on it in some way, and then prints the results as standard output.

firewall	Computer situated between your internal network and the rest of the network that filters packets as they go by according to user-specified criteria.
fragmentation	Process of dividing a packet into multiple smaller packets so that they can be sent over a communication link that only supports a smaller size.
FTP proxy	Can be configured to allow or deny specific FTP commands such as put or get.
gateway	A device that connects networks that use different communication protocols. It transfers information and converts it to a compatible format to the receiving network. See virtual private network.
HA	See high availability.
HA cluster	High availability-specific groups. Multiple secondary HA cluster Screens are managed by the primary HA cluster Screen. One Screen in an HA cluster (secondary or primary) is the active Screen that is actively filtering. Additional HA cluster Screens remain passive until one detects the failure of the active HA cluster Screen and takes over the routing and filtering of the network traffic. See high availability.
heartbeat	Periodic message sent between the machines within an HA cluster over a private network to maintain state. If the heartbeat is not detected after a specified interval and number of retries, a passive machine in the HA cluster becomes the active machine. See high availability.
high availability	Consists of one active Screen and at least one passive Screen. If the active Screen fails, a passive Screen takes over the filtering of the network traffic and other functionality of the failed firewall.
host	Name of any device on a TCP/IP network that has an IP address. In SunScreen, host is only used when referring to a source or destination of a packet.
HTTP proxy	Can be configured to ALLOW or DENY Java applets, and ActiveX controls and cookies.
ICMP	Internet Control Message Protocol. IP protocol that handles errors and control messages, to enable routers to inform other routers (or hosts) of IP routing problems or make suggestions of better routes. See ping.
IKE	See Internet Key Exchange.
Initial configuration	When installing SunScreen, the user creates, compiles, and activates a configuration named <code>Initial</code> , which enables a user to connect to the Screen where the configurations used to implement their security policy are built.

integrity	Property of ensuring that data is transmitted from the source to destination without undetected alteration.
interfaces	Describes the physical interface ports of Screen objects.
Internet Key Exchange	The Internet Key Exchange (IKE) protocol is a key management protocol standard used in conjunction with the IPSec standard.
Internet Protocol	Suite of protocols within TCP/IP used to link networks worldwide on the Internet. See IP.
IP	Internet Protocol. Network layer protocol for the Internet Protocol suite.
IPsec	An IP security feature that provides robust authentication and encryption of IP packets.
issued certificate	Certificate that is issued by a certificate authority. See self-generated certificate.
JDK	Java Development Kit. Software tools used to write Java applets or application programs.
JRE	Java Runtime Environment.
JVM	Java Virtual Machine.
key	Code for encrypting or decrypting data.
key and certificate diskette	Medium that contains the private key and certificate, and should be kept secure. The identifier for the certificate is on the label.
log browser	Facility in SunScreen administration GUI that enables the display and printing of log messages.
MAC	Message Authentication Code. (Also known as media access control, an IEEE standard.) See authentication.
Master key identifier	See MKID.
media access control	(MAC). The lower sublayer of the OSI Reference Model layer 2, the data-link layer. It controls access to a transmission medium such Token Ring, CSMA/CD, Ethernet, and the like.
message authentication code	(MAC). (Also known as media access control, an IEEE standard.) See authentication.
message transfer agent	The program responsible for delivering email messages from a mail user agent or other MTA.
MIB	Management Information Base. SNMP structure that describes the particular device being monitored. See SNMP.
MKID	Master key identifier. A component of a SKIP certificate object.
MTA	See message transfer agent.

multicast	Special form of broadcast where copies of the packet are delivered to only a subset of all possible destinations.
Name space identifier	See NSID.
NAT	See network address translation.
network address translation	Function used when packets passing through a firewall have their addresses changed (or translated) to different network addresses. Address translation can be used to translate unregistered addresses into a smaller set of registered addresses, allowing internal systems with unregistered addresses to access systems on the Internet.
network layer	Third of the seven layers in the ISO model for standardizing computer-to-computer communications.
network mask	Number used by software to separate the local subnet address from the rest of a given IP address.
node	Junction at which subsidiary parts originate or center.
nodename	Name by which the system is known to a communications network. Every system running Solaris is assigned a nodename. The nodename can be displayed using the Solaris <code>uname -n</code> command. Each Screen has a name that is normally the same as the nodename.
nonrepudiation	Property of a receiver being able to prove that the sender of a message did in fact send the message, even though the sender might later want to deny ever having sent it.
NSID	Name space identifier. Used to identify a naming scheme for a SKIP key. See key.
OLTP	Online transaction processing. Handles real-time transactions.
OSI	Open Systems Interconnection. Suite of protocols and standards sponsored by ISO to communicate data between incompatible computer systems.
OSPF	Open shortest path first. A network routing protocol.
packet	Group of information in a fixed format that is transmitted as a unit over communications lines.
packet filtering	Process to ALLOW or DENY examined traffic. See stateful packet filter.
parameter	See argument.
passive Screen	Screen in a high availability cluster that is keeping state with the active Screen but not actually passing traffic. A passive Screen will become active if the cluster's active Screen fails. See active Screen.
passphrase	Collection of characters used in a similar manner to, although longer than, password. Letters in both uppercase and lowercase can be used, as well as special characters and numbers. See password.

password	Unique string of characters that a user types as an identification code as a security measure to restrict access to computer systems and sensitive files.
peer	Any functional unit in the same layer as another entity.
PFS	Perfect Forward Secrecy. Captured packets that are decrypted cannot be used to decrypt other packets.
ping	Packet Internet Groper. Program used to test reachability of destinations by sending them an ICMP echo request and waiting for a reply. See ICMP.
plaintext	Unencrypted message.
plumb	To install and configure a network interface.
Point-to-Point Protocol	PPP (the successor to SLIP) provides router-to-router and host-to-network connections over both synchronous and asynchronous circuits.
	TCP/IP connectivity, usually for PCs over a telephone line.
policy	Named set of policy data. For example, when the SunScreen software is first installed, it configures a default policy named <code>Initial</code> .
PPP	See Point-to-Point Protocol.
primary Screen	In a high availability cluster, the Screen that controls the configuration of the cluster. In a centralized management group, the Screen that controls the configuration of the other Screens in the group. Each high availability cluster or centralized management group has exactly one primary Screen. See high availability.
private key	Corresponds to a public key and is never disclosed to the public. See secret key.
protocol	A formal description of messages to be exchanged and rules to be followed for two or more systems to exchange information.
proxies	Proxies are separate user-level applications and provide content filtering and user authentication. Proxies are used to control the content of various network services. See HTTP proxy, FTP proxy, Telnet proxy, and SMTP proxy.
pseudorandom	Pseudorandom numbers appear random but can be generated reliably on different systems or at different times.
public certificate diskette	Medium that contains only the certificate containing the public key. The identifier for the certificate is on the label.
public-key certificate	A digitally signed data structure containing a user's public key, as well as information about the time and date during which the certificate is valid.

public-key cryptography	Also known as asymmetric key cryptography. In public-key cryptosystems, everyone has two related complementary keys, a publicly revealed key and a secret key (also frequently called a private key). Each key unlocks the code that the other key makes. Knowing the public key does not help you deduce the corresponding secret key. The public key can be published and widely disseminated across a communications network. This protocol provides privacy without the need for the secure channels that a conventional cryptosystem requires.
real time	Event or system that must receive a response to some stimulus within a narrow, predictable time frame, provided that the response is not strongly dependent on highly variable system-performance parameters, such as a processor load or interface.
remote	System in another location that can be accessed through a network.
router	Intermediary device responsible for making decisions about which of several paths network (or Internet) traffic will follow.
routing mode	Routing-mode interfaces have IP addresses and perform IP routing. Routing mode requires that you subnet the network. All proxies are accessed through the transmission control protocol (TCP) and, therefore, can only run on systems with at least one interface configured in routing mode.
rules	Formulas that define a security policy in terms of the common data objects for SunScreen. Policy data include filtering rules, NAT rules, and administration access rules.
Screen-specific objects	Data objects relevant to the policies of one Screen. See common objectscommon objects.
SDNS	Secure Data Network Service.
secondary Screen	Screen that receives its configuration from a primary Screen. Normally, no administration is performed on a secondary Screen. A secondary Screen does, however, maintain its own logs and status, which can be examined. See high availability.
secret key	Corresponds to a public key and is never disclosed to the public. See private key.
securID	Software to verify authentication requests and centrally administer authentication policies for enterprise networks. See also ACE/Server.
self-generated certificate	See self-signed certificate and UDH certificate. Compare with issued certificate.
self-signed certificate	A digitally signed collection of data, whose content can be checked for authenticity, and optionally used to check the authenticity of other digitally signed collections (issued certificate).

In SKIP, the CA certificates are self-signed. Obtained out-of-band, they are used as the basis for issued certificate from a CA, no matter how they are obtained

session key	Common cryptographic component to encrypt each individual conversation between two people with a separate key.
SET	Secure Electronic Transaction. Protocol that is an emerging standard for Internet bank card transactions.
shell	Program within which a user communicates with the operating system.
SKIP	Simple Key-Management for Internet Protocols. IP-layer encryption package integrated into SunScreen, which provides a system with the ability to encrypt any protocol within the TCP/IP suite efficiently. Once installed, systems running SunScreen SKIP can encrypt all traffic to any SKIP-enabled product, including SunScreen products.
SMTP	Simple Mail Transfer Protocol. Used on the Internet to route email.
SMTP proxy	TCP/IP protocol that sends messages from one computer to another on a network and is used on the Internet to route email.
SNMP	Simple Network Management Protocol. Network management protocol that enables a user to monitor and configure network hosts remotely.
snoop	Sun Microsystems, Inc. UNIX utility that captures packets from the network and displays their contents.
source code	Uncompiled version of a program written in a language such as C, C++, or Java. The source code must be translated to machine language by a program (the compiler) before the computer can execute the program.
stateful packet filter	Packet filter that bases its decision to allow or deny the packet using both the data in the packet and information (that is, state) saved from previous packets or events. A stateful packet filter has memory of past events and packets.
stateless packet filter	Packet filter that bases its decision to allow or deny a packet using only the data in that packet. A stateless packet filter has no memory of past events and packets.
static translation	Address translation that provides fixed translation between an external address and a private (possibly unregistered) address. It provides a way for external hosts to initiate connections to internal hosts without actually using an external address. See network address translation.
stealth mode	Stealth-mode interfaces do not have IP addresses. They bridge the MAC layer. Stealth mode interfaces partition an existing single network and, consequently, do not permit you to subnet the network.

	If all of your interfaces are in stealth mode, SunScreen offers optional hardening of the OS, which removes packages and files from the Solaris operating system that are not used by SunScreen.
subnet	In the Internet Protocol, a mechanism to subdivide (registered) networks into locally defined pieces. This technique provides better use of the IP address space while minimizing routing-table complexity. See subnet mask.
subnet mask	Specifies which bits of the 32-bit IP address represent network information. The subnet mask, like an IP address, is a 32-bit binary number: a 1 is entered in each position that will be used for network information and a 0 is entered in each position that will be used as node number information. See node.
SunScreen	Name of the family of security products produced by Sun Microsystems, Inc.
SunScreen SKIP	See SKIP.
TCP	See Transmission Control Protocol .
TCP/IP	Transmission Control Protocol/Internet Protocol. Protocol suite originally developed by the Department for Defense for the Internet. It is also called the Internet protocol suite. SunOS networks run on TCP/IP by default.
Telnet proxy	Enables users of one host to log into a remote host and interact as normal terminal users of that host.
traffic analysis	Analysis of network traffic flow for the purpose of deducing information such as frequency of transmission, the identities of the conversing parties, sizes of packets, flow identifiers used, and the like.
Transmission Control Protocol	The protocol within TCP/IP that governs breaking data messages into packets that are sent using IP, reassembling these packets into the complete message, and verifying the reassembled message as the same as the original data message.
tunnel address	Destination address on the outer (unencrypted) IP packet to which tunnel packets are sent. Generally used for encrypted gateways where the IP address of the host serves as the intermediary for any or all hosts on a network whose topography must remain unknown or hidden from the rest of the world.
tunneling	Process of encrypting an entire IP packet, and wrapping it in another (unencrypted) IP packet. The source and destination addresses on the inner and outer packets may be different.
UDH certificate	Unsigned Diffie-Hellman certificate. UDH public value can be used when entities are named using the message digest of their DH public value, and these names are securely communicated. This term is now mostly replaced by self-signed certificate. See certificate identifier (ID).

UDP	User Datagram Protocol. All CDP communication uses UDP.
unicast	Packet sent to a single destination. Compare broadcast, multicast.
version	Manner in which a policy's historical versions are preserved.
virtual private network	<p>A network with the appearance and functionality of a regular network, but which is really like a private network within a public one.</p> <p>The use of encryption in the lower protocol layers provides a secure connection through an otherwise insecure network, typically the Internet. VPNs are generally cheaper than real private networks using private lines but rely on having the same encryption system at both ends. The encryption may be performed by firewall software or possibly by routers.</p>
VPN	See virtual private network.
VPN gateway	See virtual private network.

Index

Numbers and Symbols

* service, 282

5-tuple, 111

A

access control, 22, 143

 defining rules, 90

 overview, 21

 packet filtering rules, 29

ACE

 SecurID, 150

 stub client, 150

ACE/Agent, 149, 150, 151

ACE/Client, 149, 150

acemaster, 154

ACE/Server, 132, 134, 143, 148, 149, 150, 152, 153, 154, 155, 157

Ace/Server, 151

action, 46

 ICMP message, 46

 SNMP message, 46

address

 gateway object, 98

address object

 definition, 60

 group, 62

 limitation, 63

 host, 61

 modifying address note, 63

 multiple Screens, 87

address object (*continued*)

 range, 61

address set

 valid, 69

addresses

 hiding

 tunneling, 106

admin interface, 69, 72

ADMIN interface

 SunScreen Lite, 27

administration graphical user interface. See administration GUI, 21

administration GUI, 22

 browser support, 31

 command-line user interface, 221

 configuration editor

See also configuration editor

 end-system SKIP, 31

 graphical user interface, 31

 interoperability with command line, 219

 overview, 21

 version number, 52

Administration Station

 components, 33

 description of, 21

 remote administration, 84

administrative user, 135

 authentication, 130

AH (authentication header)

 IPsec/IKE, 41

ah service, 283

archie service, 283

- authentication, 22, 166
 - external users, 129
 - internal users, 129
 - IPsec, 43
 - IPsec/IKE, 41
 - MD5, 43
 - overview, 21
 - SHA-1, 43
- authentication events, 196
- authentication header (AH)
 - IPsec/IKE, 41
- authorized user, 130
 - authentication, 129, 142
 - authentication processing logic, 134
 - creating, 132
 - defining object, 130
 - example
 - create object, 133
 - create object defining SunScreen, 133
 - create simple-text object, 134
 - display existing object, 132
 - display object names, 134
 - display objects, 134
 - RADIUS details, 143

B

- BROADCAST, 275
- broadcast traffic, 285
 - addbroadcast, 286
 - new service, 286

C

- CA, *See* certificate authority
- CA issued note, 90
- caution
 - dynamic NAT, 291
- centralized management
 - common object for, 86
 - screen objects, 86
- centralized management group
 - concepts, 85
 - logs, 86
 - primary Screen, 85, 87

- centralized management group (*continued*)
 - secondary Screen, 85
 - setting rules, 87
 - SunScreen Lite, 26
- certdb, 224
 - IKE & SKIP databases, 228
 - ssadm subcommand, 228
- certificate
 - associate MKID, 80
 - gateway object, 99
 - IKE, 42
 - MKID, 80
 - Sun CA, 90
- certificate authority, 42
- certificate object
 - associating with a Screen, 80
 - definition, 79
 - group, 80
 - optional description, 80
 - single, 80
 - unique name, 80
- certlocal, 224
 - local IKE & SKIP databases, 229
 - ssadm subcommand, 229
- certrldb, 224
 - certificate revocation lists, 229
 - ssadm subcommand, 229
- character
 - forbidden, 73, 77, 78, 79, 136
- ciphertext message
 - proxies, 89
- command
 - harden, 33
- command line
 - editor, 22
- command line user interface
 - configuration editor, 220
- command-line user interface
 - accessing Screen, 220
 - administration GUI, 81, 221
 - reference, 219
- commands
 - configuration editor, 242
 - SunScreen SKIP commands, 240
 - UNIX, 221
 - unsupported, 237
- common object, 55

commands (*continued*)

- address, 60
- authorized user, 73, 130
- automatically saved, 55
- certificate, 79
- data object, 37
- database, 55
- interface, 68, 71
- jar hash, 79
- jar signature, 78
- multiple Screens, 86
- not automatically saved, 56
- policy rules, 55
- proxyuser, 77
- require saving, 55
- Screen, 64
- service, 56

common objects

- administrative user, 135
- automatically saved, 163
- proxy user, 135

compatibility

- SKIP, 26
- SunScreen, 25

complete valid address set, 69

components

- Administration Station, 21, 33
- Screen, 21, 33, 83

configuration

- common object, 37
- security policy, 37

configuration editor

- See also* administration GUI, 220
- command line, 22
- create controlling objects, 240
- data model, 240
- object types, 242

content filtering, 166

content scanning

- VirusWall, 181

controlling objects

- creating, 240

CoolTalk service, 283

cryptography

- authentication, 90
- network layer note, 90
- privacy, 90

cryptography (*continued*)

- public-key, 39, 90
- shared-key, 39, 90

D

data algorithm

- gateway object, 99

data model

- configuration editor, 240

data object

- common object, 37

data protection

- IPsec/IKE, 41

database

- common object, 55

decrypting packets, 39, 90

decryption

- function details, 89
- IPsec/IKE, 41

defining

- VPN, 98

description, 21

- gateway object, 99

detail, 193

Diffie-Hellman, *See also* IKE

disabled interface, 69

discriminator, 295

- port, 275
- RPC number, 275
- type, 275

distinguished name

- IKE, 80
- single certificate, 80
- SKIP, 80

dns service, 283

dns state engine, 296

documentation, 27

- location of PDF files, 27

DSS signature, *See* IKE

dynamic NAT

- caution, 291
- trusted Solaris, 291

E

- editor
 - command line, 22
 - GUI, 22
- email configuration
 - SMTP proxy, 174
- encapsulating security payload (ESP)
 - IPsec/IKE, 41
- encrypting packets, 39, 90
- encryption, 22
 - DES, 43
 - devices
 - when required, 106
 - function details, 89
 - IPsec, 43
 - IPsec/IKE, 41
 - overview, 21
 - proxies, 89
 - public-key cryptography, 39, 90
 - shared-key cryptography, 39, 90
 - SKIP, 26
 - SunScreen Lite, 26
 - triple-DES, 43
- error message
 - adminuser, 220
 - authuser, 220
 - lock not held, 220
 - logmacro, 220
 - proxyuser, 220
 - save log macro, 208
 - vars, 220
- error messages, 313, 321
 - logged packet reasons, 323, 327
 - ssadm activate, 309
 - ssadm edit, 309
 - ssadm lock, 309, 321
- ESP (encapsulating security payload)
 - IPsec/IKE, 41
- esp service, 283
- ether
 - state engine, 296
- ethernet
 - type field, 298
- ethernet 802.2, 297
- ethernet 802.3, 297
- ethernet II, 297
- ethernet SNAP, 297

- external users
 - authentication of, 129

F

- failover protection
 - HA, 32
- feature
 - HA, 32, 39
 - NAT, 32, 37, 105
- filtering, 166
 - packets, 37
- FTP proxy
 - anonymous FTP, 164
 - controlling site access, 163
 - destination address, 163
 - example
 - display variable, 165
 - primary Screen, 165
 - functions, 163
 - global version, 165
 - limiting access to ftp commands, 163
 - source address, 163
- ftp service, 284
- ftp state engine, 284, 299
 - PASV mode, 299
- function
 - stateful packet filtering, 46

G

- gateway
 - VPN, 60
- gateway object
 - address, 98
 - certificate, 99
 - data algorithm, 99
 - description, 99
 - key algorithm, 99
 - MAC algorithm, 99
 - name, 98
 - rule index, 98
 - tunnel address, 99
 - VPN, 98

- getpeerinfo
 - tsolpeerinfo invokes, 290
- global log limiter, 196
- graphical user interface, 22, 31
 - administration GUI, 31
 - install wizard, 31
 - skiptool GUI, 31
- group, 290
- GUI
 - administration, 31
 - skiptool, 31
- GUI editor, 22

H

HA

- active Screen, 120
- automatic disconnection, 126
- communication in, 125
- configuring, 126
- definition of, 119
- disrupted connections, 126
- event log, 190
- failover, 126
- failover protection, 32
- failure of primary Screen, 125
- function details, 39, 119
- limitations, 126
- lost connections, 126
- NAT, 123
- overview, 21
- passive Screens, 39, 120
- primary Screen, 120
- reinstate Screen, 126
- remote administration, 122
- routing mode, 121
- secondary Screen, 120
- setting an HA cluster, 120
- Solaris settings, 120
- state information limitations, 126
- SunScreen Lite, 27

HA cluster, 121

- communicating with `ftp` and `telnet` to members of, 127
- communication between members, 126
- forcing failover, 126

HA cluster (*continued*)

- function details, 39
- hubs necessary for, 127
- mirror configuration, 122
- mirror state, 122
- naming Screen, 127
- network interface, 120
- non-switching hub, 71
- stealth interface, 71

HA interface, 69

- SunScreen Lite, 27

harden command, 33

hardening OS

- optional, 33
- stealth mode, 33

help

- documentation, 27
- man pages, 27
- online, 27

high availability

- non-switching hub, 71
- stealth interface, 71

high availability. See HA, 21

HTTP proxy

- defining source address, 166
- example
 - display variable, 167
- filtering content, 166
- filtering Java applets, 166
- filtering restrictions, 167
- ftp access, 168
- functions, 166
- limitations, 173
- NAT implementation, 166
- operation, 171
- prevent access, 168
- restrict Web content, 168
- SSL support, 166
- useful in implementing NAT, 166
- using Java, 168
- VirusWall scanning, 166

hub

- non-switching
 - HA cluster, 71
 - stealth mode, 71

I

- ICMP messages, 48
- ICMP packets, 285
- icmp service, 285
- icmp state engine, 285, 300
- IETF standard, 41
- IKE
 - See also* IPsec, 22, 41
 - certdb, 228
 - certificate, 42
 - certlocal, 229
 - certlrb, 229
 - IPsec SA, 42
 - IPsec security association, 42
 - signature, 42
- IKE peers, 42
- incomplete valid address set, 69
- individual servers
 - SunScreen Lite, 26
- installation
 - requirements, 23
- integrity validation, 41
- interface, 21
 - admin, 72
 - HA, 72
 - HA cluster network, 120
 - mixed routing and stealth, 72
 - modes, 21
 - routing, 21, 70, 71
 - routing mode, 21, 32
 - stealth mode, 21
- interface object, 68
- interface objects
 - single Screen, 87
- interface type
 - admin, 69
 - disabled, 69
 - HA, 69
 - routing, 69
 - stealth, 69
- interfaces
 - SunScreen Lite, 27
- internal users
 - authentication of, 129
- Internet Key Exchange, *See* IKE
- Internet Key Exchange (IKE), 41
- Internet Protocol security (IPsec), 41

- IP address
 - defining rules, 90
- IP addresses, 33
- ip all service, 286
- ip forward service, 286
- ip mobile service, 286
- IP protocol, 41
- (IP Security Architecture/ Internet Key Exchange. *See* IPsec/IKE, 22
- ip tunnel service, 286
- ipfwd state engine, 300
- ipmobile state engine, 300
- IPsec, 22, 42
 - authentication, 43
 - configuration, 43
 - DES, 43
 - encryption, 43
 - MD5, 43
 - SHA-1, 43
 - triple-DES, 43
- IPsec SA, *See also* IKE
- ipsec service, 287
- IPsec/IKE, 22, 41
 - AH (authentication header), 41
 - authentication, 41
 - data protection, 41
 - decryption, 41
 - encryption, 41
 - ESP (encapsulating security payload), 41
 - integrity validation, 41
- IPsec/IKE and SKIP, 41
- iptunnel state engine, 301
- ipv6 tunnel service, 287
- isakmp service, 287

J

- Jar hashes
 - VJM, 173
- Jar hashes and signatures
 - JVM, 173
- Jar signatures, 173
- Java
 - plug-in
 - installation instructions, 25
 - Solaris, 25

Java, plug-in (*continued*)
 Windows, 25
Java Virtual Machine, see JVM, 172
JVM, 172
 Jar hashes, 173
 Jar hashes and signatures, 173

K

key algorithm
 gateway object, 99
key manager
 SunScreen SKIP, 91
keys
 issued, 90

L

local administration
 concepts, 34, 84
 overview, 22
 routing mode Screen, 84
 routing-mode Screen, 34
log
 altering size, 192
 automated centrally managed group, 196
 automated management, 196
 automated postprocessing logs, 196
 binary records, 197
 bridging macros, 207
 centralized management group, 86, 191
 common optional attributes, 206
 embedded string filters, 199
 examining, 39, 189
 example, 197
 clear, 197
 clear log, 198
 debugging, 196
 defining specific macro, 208
 display global default, 191
 display global log limiter, 195
 display macro definition, 208
 display Screen definitions, 210
 display Screen names, 210
 display size specific Screen, 192

log, example (*continued*)
 display specific macro definition, 210
 displaying log statistics, 198
 expanding given macro, 213
 expanding log macro, 212
 get_and_clear automatically, 197
 get_and_clear log, 198
 logapp operand, 206
 logsev operand, 205
 processing local file log record, 199
 processing records, 199
 retrieving items from current log, 213
 setting size specific Screen, 192
extended events, 192, 193
extended log event enhancements, 205
extended log events, 204, 206
filtering macros, 206
filtering mechanisms, 205
filtering Screen logs, 199
general event type enhancements, 204
get_and_clear operation, 197
global default size, 191
group-Screen installations, 207
HA, 190
 HA cluster, 191
 installation, 191
 limiter variables, 195
 limiters, 195
 list, 192
 listing macros, 210
 local macros, 207
 locations, 190
 logdump extensions, 200
 logged network packet enhancements, 203
 logging server, 198
 macro expansion, 212
 macros, 206
 macros registry, 206
 manual management, 196
 naming macros, 207
 network session, 192, 193
 network traffic, 192
 packet filtering, 196
 primary Screen log file size, 191
 propagating limiters, 195
 reason why packet logged, 200
 retrieval and clearing, 196

- log, example (*continued*)
 - secondary Screen log file size, 191
 - session summary events, 204
 - snoop, 205
 - specific Screen, 86
 - statistics, 198
 - traffic size, 190
 - using log macros, 212
 - who cleared log, 197
- log browsing
 - active Screen, 199
 - administration GUI, 199
- log statistics
 - administration GUI, 198
 - command-line retrieval, 198
- LOG_DETAIL, 193
- logdump
 - derived from snoop, 272
- logged packet reasons, 323, 327
 - why codes, 323
- logging, 39
 - packet logging
 - detail, 193
 - none, 193
 - sessions, 193
 - spoofed packet, 70
- logmacro
 - save error message, 208
- LOG_NONE, 193
- LOG_SESSION, 193
- LOG_SUMMARY, 193

M

- MAC address, 122
- MAC algorithm
 - gateway object, 99
 - VPN gateway, 103
- macros
 - log filtering, 206
- man page
 - ssadm logdump, 272
- man pages
 - help, 27
 - IKE, 27
 - SKIP, 27

- man pages (*continued*)
 - Solaris, 27
 - SunScreen, 27
- master key identifier, *See* MKID
- message authentication code, *See also* MAC
 - address
- message transfer agent, *See* see MTA
- mirror configuration
 - HA cluster, 122
- mirror state
 - HA cluster, 122
- mixed-mode
 - routing and stealth, 69
- MKID
 - SKIP certificate, 80
- multiple Screens
 - address object, 87
 - common object, 86
 - policy rules for, 86

N

- name
 - forbidden characters, 73, 77, 78, 79, 136
 - gateway object, 98
 - VPN, 98
- name space identifier, *See* NSID
- naming conventions, 63
- NAT
 - and tunneling, 106
 - caution, 291
 - configuration, 106
 - demilitarized zone, 113
 - dynamic, 109
 - example mappings, 112
 - function details, 105
 - HA, 123
 - mapping collisions, 110
 - ordered translations, 105
 - sequence, 46
 - site mappings, 116
 - stateful, 106
 - static, 107
 - SunScreen Lite, 27
- network packet traffic, 204

- network security policy
 - setting up, 57
- network services
 - service groups, 292
- network topology
 - security policy, 33, 83
- new service
 - ip, 286
 - ip fwd, 286
 - ip mobile, 286
 - ip tunnel, 286
- nfs readonly service, 287
- nis state engine, 301
- no logging, 193
- non-switching hub
 - HA cluster, 71
 - stealth interface, 71
- NSID
 - SKIP certificate, 80
- ntp
 - service, 287
 - state engine, 287, 301
- ntp service, 287

O

- object types
 - configuration editor, 242
 - named, 37
 - ordered, 37
- online help, 27
- ordered rules sequence, 48
- overview
 - access control, 21
 - administration GUI, 21
 - authentication, 21
 - encryption, 21
 - HA, 21
 - local administration, 22
 - public-key encryption, 21
 - remote administration, 22

P

- packet
 - filtering, 37
 - IP screening guidelines, 285
- packet filtering, 37
 - routing—mode sequence, 46
 - set up rules, 29
 - state engine, 21
 - stateful service rules, 58
 - stealth—mode sequence, 46
- packet logging, 39, 189
- packets
 - ALLOW rule, 47
 - checking size, 90
 - concatenated, 91
 - creating, 91
 - decrypting, 39, 90
 - encapsulated, 90
 - encrypting, 39, 90
 - filtering, 286
 - fragmentation, 90
 - ICMP, 285
 - ICMP screening guidelines, 284
 - logged error messages, 309
 - logging, 39, 189
 - passing RIP, 288
 - replacing addresses, 91
 - restoring original, 91
 - transmission, 90
 - tunneling, 91
 - VPN, 38, 92
- parameters, 295
- PASV mode (FTP), 299
- PDF files, 27
- PIN
 - SecurID, 153
- ping state engine, 302
- plaintext message
 - proxies, 89
- pmap_nis state engine, 302
- pmap_tcp state engine, 302
- pmap_udp state engine, 303
- policy, 31
 - currently active, 52
 - new version, 53
 - older version, 53
 - rules with multiple Screens, 86

- policy (*continued*)
 - version or history, 31
- policy rules
 - function details, 48
 - ordered, 48
 - rule syntax, 50
- policy versions, 52
- pre-shared key, *See* IKE
- primary Screen, 122
 - common object, 85, 87
 - HA, 120
 - screen objects, 86
- primary Screen in centralized management
 - SunScreen Lite, 27
- primary Screens
 - centralized management group, 85, 87
- protocol
 - IP, 41
- proxies, 27, 40
 - activate policy, 40
 - ciphertext message, 89
 - client software, 40
 - content filtering, 40
 - encryption, 89
 - extend, 129
 - FTP protocol, 40
 - FTP proxies, 143
 - HTTP protocol, 40
 - plaintext message, 89
 - regulate, 129
 - RSA Security ACE/Server, 143
 - server software, 40
 - setting rules, 40
 - SMTP protocol, 40
 - SunScreen Lite, 27
 - system configurations, 40
 - TCP protocol, 161
 - Telnet protocol, 40
 - UDP protocol, 161
 - user authentication, 40
 - variables RADIUS client protocol, 147
- proxy
 - activate policy, 159
 - client software, 159
 - content filtering, 159
 - DNS configuration, 178
 - establish proxy user authenticity, 162

- proxy (*continued*)
 - example
 - session illustration, 164
 - FTP connection, 165
 - FTP protocol, 159, 163
 - FTP proxy collateral mapping, 162
 - how proxies work, 160
 - HTTP
 - VirusWall, 181
 - HTTP protocol, 159, 166
 - JAR hashes, 173
 - limitations, 162
 - locate proxy user authenticity rule, 162
 - multithreaded program, 160
 - MX records, 178
 - policy rule matching, 161
 - protocols, 159
 - proxy user anonymous, 164
 - SecurID PIN server, 159
 - server software, 159
 - setting rules, 159
 - SMTP, 181
 - SMTP protocol, 159, 174
 - system configurations, 159
 - Telnet protocol, 179
 - Telnet protocol, 159
 - user authentication, 159, 162
- proxy user
 - authentication, 129
 - creating object, 138
 - defining object, 137
 - example
 - add GROUP members, 140
 - create GROUP object, 139
 - create SIMPLE object, 139
 - display all names, 140
 - display all objects, 140
 - display objects, 139
 - remove GROUP object, 140
 - FTP proxies, 141
 - GROUP object, 137
 - GROUP objects, 135
 - login, 141
 - Login page, 141
 - object definition, 130
 - RADIUS, 135
 - RADIUS access to LDAP, 135

- proxy user, example (*continued*)
 - RADIUS LDAP stored in SDS, 135
 - SecurID, 135
 - SIMPLE null authentication, 141
 - SIMPLE object, 137
 - SIMPLE objects, 135
 - SPECIAL external authentication
 - method, 142
 - special objects, 135
 - Telnet proxies, 141
- public key certificate
 - X509, 42
- public-key cryptography, 39, 90
- public-key encryption
 - overview, 21

R

RADIUS

- example
 - create address objects, 146
 - create node secret, 146
 - create rule, 146
 - create variables, 146
- multiple-Screen installations, 145
- prefigured parameters, 144
- requestor, 144
- response time, 147
- server port, 147
- testing, 148
- testing by SDS, 148
- testing by SecurID, 148
- typical configuration, 146
- UDP datagrams, 144
- user authentication details, 143
- variables, 144, 147

RealAudio, 287

realaudio service, 287, 303

realaudio state engine, 303

remote administration

- ADMIN interface, 123
- Administration Station, 84
- concepts, 34, 84
- HA, 122
- overview, 22
- Screen, 84

remote administration (*continued*)

- SunScreen Lite, 26
- remote shell (rsh), 304
- remote-access server, 31
- requirements
 - hardware, 23
 - installation, 23
 - software, 23
- rip service
 - RIP packets, 288
- routing and stealth
 - mixed-mode, 69
- routing information protocol
 - RIP, 288
- routing interface, 69, 70
- routing mode, 21, 32
 - HA limitations, 126
 - interface, 32
 - limitations, 126
 - remote-access server, 31
 - subdividing a network, 21
 - traditional firewall, 31
 - virtual interface, 21
- rpc service, 288
- rpc_tcp state engine, 303
- rpc_udp state engine, 304
- RSA encryption, *See* IKE
- RSA signature, *See* IKE
- RSA-ENCRYPTION, 43, 95
- rsh state engine, 304
 - remote shell sessions, 304
- rule
 - ALLOW, 47
 - DENY, 47
- rule index
 - gateway object, 98

S

SA (security association)

- IPsec/IKE, 41

sample network map, 30

Screen

- active HA Screen, 39
- components, 33, 83
- configuration objects, 85

- Screen (*continued*)
 - HA limitations, 126
 - managing multiple Screens, 34
 - multiple management, 84
 - passive, 39
 - primary, 122
 - reinstate, 126
 - remote administration, 84
 - remote headless, 83
 - secondary, 122
- Screen description of, 21
- screen object
 - centralized management, 86
 - primary Screen, 86
 - secondary Screen, 86
- screening guidelines
 - ICMP packets, 284
 - IP packets, 285
- secondary Screen, 122
 - administration capabilities, 86
 - HA, 120
 - screen objects, 86
- secondary Screens
 - centralized management group, 85
- SecurID
 - access paths, 152
 - ACE, 150
 - ACE/Agent installation, 151
 - example
 - token PIN establishment, 156
 - example configuration, 154
 - example create registry address, 155
 - example stub client configuration, 155
 - stub client, 150
 - stub client location, 151
 - token PIN, 153
 - typical authentication, 149
 - UDP and TCP protocols, 157
 - use caution in deployment, 157
- security association (SA)
 - IPsec/IKE, 41
- security considerations, 29
- security network
 - sample network map, 30
- security parameters index (SPI), 41
- security policy
 - Initial, 37
- security policy (*continued*)
 - network topology, 33, 83
 - ordered policy rules, 48
 - policy objects, 37
 - security decisions, 36
- service
 - *, 282
 - ah, 283
 - archie, 283
 - CoolTalk, 283
 - dns, 283
 - entries for ports, 57
 - esp, 283
 - ftp, 284
 - icmp, 285
 - ip all, 286
 - ip mobile, 286
 - ipsec, 287
 - ipv6 tunnel, 287
 - isakmp, 287
 - network service groups, 292
 - nfs readonly, 287
 - ntp, 287
 - predefined, 56
 - realaudio, 287
 - rip, 288
 - rpc, 288
 - single
 - broadcast filter, 58
 - reverse filter, 58
 - smtp, 289
 - sqlnet, 289
 - TCP, 289
 - tcp all, 292
- service
 - traceroute, 290
- service
 - tsolpeerinfo, 290
 - udp, 291
 - VDOLive, 291
 - www, 292
- service object
 - definition, 56
 - group, 57
 - creating new service, 59
 - definition, 59
 - modifying, 59

- service object, group (*continued*)
 - predefined, 59
 - single, 57
 - creating new, 57
 - forward filter, 58
 - keyword, 58
 - modifying, 57
 - port filter, 58
 - state engine, 57
- services
 - discriminator, 275
 - realaudio state engine, 303
 - standard, 275
 - state engine, 275
- session logging, 193
- shared-key cryptography, 39, 90
- shell commands, 221
- signature
 - IKE, 42
- single Screen
 - interface objects, 87
- SKIP, 22
 - certlocal, 228, 229
 - compatibility, 26
 - encryption, 26
 - RC2 limitation, 25
 - SunScreen Lite, 26
- SKIP and IPsec/IKE, 41
- SKIP certificate
 - NSID, 80
- skiptool GUI
 - encryption of administration commands, 90
 - graphical user interface, 31
- small work groups
 - SunScreen Lite, 26
- SMTP proxy
 - create rules, 179
 - email configuration, 174
 - email configuration issues, 178
 - example
 - add restrictions, 177
 - define address group, 177
 - define relay restrictors, 177
 - define spam restrictors, 176
 - display restrictors, 176
 - displaying spam restrictors, 175
 - email rule, 178
- SMTP proxy, example (*continued*)
 - remove restriction, 176, 177
 - functions, 174
 - MTA filtering, 179
 - operation, 174
 - rules, 179
 - spam
 - control, 175
 - VirusWall scanning, 174
- smtp service, 289
- SNMP
 - alerts, 66
 - IP addresses, 66
 - receivers, 66
 - time status indicator, 66
 - timer interval, 66
- SNMP traps, 48
 - supported, 66
- snoop, 205
- snoop
 - logdump derived from, 272
- snoop program, 38, 92, 272
- Solaris
 - Trusted Solaris 8 for the SPARC
 - platform, 30
- Solaris, compatible versions for the SPARC and
 - Intel platforms, 30
- Solaris IPsec, *See* IPsec
- spam
 - control, 175
 - restrictors
 - defining, 175
 - syntactic forms, 175
 - restrictors
 - displaying, 175
 - working with, 175
- SPI (security parameters index), 41
- spoof protection, 69
- SQL *Net protocol, 289
- sqlnet state engine, 289
- ssadm
 - certdb subcommand, 228
 - certlocal subcommand, 229
 - certrlldb subcommand, 229
- ssadm logdump
 - man page, 272

- standard
 - IETF, 41
- star service, 282
- state engine
 - characteristics, 295
 - connection management, 295
 - definition, 57
 - discriminator, 275
 - discriminator value, 295
 - discriminators, 295
 - dns, 296
 - ether, 296
 - ftp, 295, 299
 - icmp, 300
 - ip, 300
 - ipfwd, 300
 - ipmobile, 300
 - iptunnel, 301
 - new service, 286
 - nis, 301
 - ntp, 287, 301
 - parameters, 295
 - ping, 302
 - pmap_nis, 302
 - pmap_tcp, 302
 - pmap_udp, 303
 - precedence level, 295
 - realaudio, 303
 - rpc_tcp, 303
 - rpc_udp, 304
 - rsh, 304
 - services, 275
 - tcp, 295, 305
 - tcpall, 306
 - udp, 306
 - udpall, 307
 - udp_datagram, 308
 - udp_stateless, 308
- state engines, 295
- state information
 - HA limitations, 126
- stateful packet filtering, 37
 - details, 46
- statistics
 - log file, 198
- stealth, 21
- stealth interface, 69
- state engine (*continued*)
 - HA cluster, 71
 - high availability, 71
 - non-switching hub, 71
- STEALTH interface
 - SunScreen Lite, 27
- stealth mode, 32
 - acts as a bridge, 21
 - description, 21
 - hardening OS, 33
 - interface, 33
 - SunScreen Lite, 27
- summary
 - packet logging
 - summary, 193
- SunScreen
 - command compatibility, 26
 - compatibility, 25
 - configuration editor, 215
 - error messages, 309
 - example
 - continue adding SecurID rules, 155
 - how it works, 30
 - migration from SunScreen EFS, Release 2.0, 215
 - migration from SunScreen SPF-200, 215
 - upgrading, 26
- SunScreen 3.2
 - prerequisites, 15
 - resources, 17
- SunScreen and SunScreen Lite
 - common features SunScreen Lite and SunScreen
 - common features, 26
- SunScreen compared with SunScreen Lite, 26
- SunScreen EFS 1.1, 26
- SunScreen Lite, 26, 27
 - ADMIN interface, 27
 - centralized management group, 26
 - encryption, 26
 - HA, 27
 - HA interface, 27
 - individual servers, 26
 - interfaces, 27
 - limitations, 27
 - NAT, 27
 - number of interfaces, 27

- SunScreen Lite (*continued*)
 - primary Screen in a centralized management, 27
 - remote administration, 26
 - SKIIP, 26
 - small work groups, 26
 - STEALTH interface, 27
 - stealth mode, 27
 - time-of-day rules, 27
- SunScreen Lite compared with SunScreen, 26
- SunScreen SKIP
 - commands, 240
 - end-system SKIP, 31
 - header, 90
 - key manager, 91
 - limitations note, 91
 - log, 190
- SunScreen SKIP. See SKIP, 22

T

- tcp all service, 292
- TCP service, 289
- tcp state engine, 305
- tcpall state engine, 306
- tcp_keepalive, 305
- telnet
 - trusted Solaris system, 290
- Telnet proxy
 - example
 - SunScreen SKIP, 181
 - functions, 179
 - operation, 179
 - other issues, 180
 - request user name, 179
 - TCP, 180
- time objects, 27
 - SunScreen Lite, 27
- time-based rules
 - function details, 75
- time-of-day rules
 - SunScreen Lite, 27
- traceroute service, 290
- traditional firewall, 31
- traffic key
 - generated, 91

- traffic log size, 190
- troubleshooting
 - access to console, 272
 - gathering information, 273
 - printing debug information, 272
 - ss_debug_level, 272
- trusted Solaris
 - dynamic NAT, 291
 - telnet example, 290
 - tsolpeerinfo, 290
- Trusted Solaris 8 for the SPARC platform, 30
- tsolpeerinfo
 - getpeerinfo invocation, 290
- tsolpeerinfo service, 290
- tunnel address
 - gateway object, 99
- tunneling
 - and VPN, 106
 - function details, 38, 91
 - hiding addresses, 91
 - packets, 91

U

- UDP
 - traceroute service, 290
- udp service, 291
- udp state engine, 306
- udpall state engine, 307
- udp_datagram state engine, 308
- udp_stateless state engine, 308
- UNIX commands, 221
- unsupported commands, 237
- upgrading, 26, 31
 - Solaris support, 31
 - Unicode internationalization note, 31
- upgrading from SunScreen EFS 1.1, 26
- user authentication, 166
 - administrative user, 130
 - authuser, 129
 - function details, 129
 - proxy user, 129
- users
 - external, 129
 - internal, 129

V

- valid address set, 69
 - complete, 69
 - incomplete, 69
- VDOLive service, 291
- version
 - new policy, 53
 - older policy, 53
 - policy, 52
- version number, 52
 - administration GUI, 52
 - historical, 52
 - policy versions, 52
- virtual private network, *See* VPN
- virtual private network. *See* VPN, 22
- VirusWall
 - content scanning, 181
 - filtering scripts and applets, 166
 - HTTP proxy, 166, 181
 - mail viruses, 174
 - SMTP proxy, 181
- VJM
 - Jar signatures, 173
- VPN, 26, 100
 - adding a rule, 101
 - defining, 98
 - encryption devices required, 106
 - example configuration, 100
 - function details, 38, 91
 - gateway object, 98
 - limitations, 103
 - name, 98
 - overview, 21
 - setting up, 38, 92
 - SunScreen Lite, 26
 - tunneling data, 92
- VPN gateway, 60
- vpngateway, 95

W

- why codes
 - logged packet reasons, 323
- www service, 292