Reference Manual

SunScreen EFS™ 3.0

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A. 650-960-1300

Part No. 805-7746-11
August 1999, Revision B

Please
Recycle

Adobe PostScript

Contents

Figures

Tables

Preface

SunScreen EFS™ 3.0 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. Past SunScreen firewall product releases include EFS, SPF-100, SPF-100G and SPF-200, their respective Administration Stations, SunScreen packet screen software, and SunScreen Simple Key-Management for Internet Protocols (SKIP) encryption software. SunScreen EFS 3.0 integrates the two SunScreen firewall products.

The *SunScreen EFS Reference Manual* contains background and reference information about SunScreen EFS 3.0. Other documentation in this SunScreen EFS 3.0 documentation set includes:

- *SunScreen EFS 3.0 Release Notes*
- *SunScreen EFS 3.0 Installation Guide*
- *SunScreen EFS 3.0 Administration Guide*
- *SunScreen EFS 3.0 QuickStart Card*
- *SunScreen SKIP 1.5 User's Guide*

Who Should Use This Book

The *SunScreen EFS 3.0 Reference Manual* 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.

SunScreen EFS 3.0 Requisites

You need to have the following tasks completed before you install and administer your SunScreen EFS 3.0:

- Become familiar with the SunScreen EFS 3.0 guides:
  - *SunScreen EFS 3.0 Release Notes* (PN 805-7749-11)
  - *SunScreen EFS 3.0 Installation Guide* (PN 805-7744-11)
  - *SunScreen EFS 3.0 Administration Guide* (PN 805-7745-11)
  - *SunScreen SKIP 1.5 User's Guide*
- Ensure that your system is running either Solaris 2.6 or Solaris 7.
- Ensure that your system running SunScreen EFS 3.0 is secure.
- List the network services by location (configuration matrix) allowed and disallowed per location used to establish rules.

How This Book Is Organized

The *SunScreen EFS 3.0 Reference Manual* contains the following chapters and appendices:

- **Chapter 1, "SunScreen EFS 3.0 Overview**," provides a brief overview of the SunScreen EFS 3.0 product, including such topics as operating system and hardware requirements, and major features.
- **Chapter 2, "SunScreen EFS 3.0 Concepts**," discusses the concepts and functions of SunScreen EFS 3.0.
- **Chapter 3, "Graphical User Interface Reference**," explains the SunScreen EFS 3.0 graphical user interface (GUI), including navigation, page descriptions, and field descriptions.
- **Chapter 4, "SunScreen Logs**," describes SunScreen EFS 3.0 packet logging.
- **Chapter 5, "Proxies**," describes the proxies supported by SunScreen EFS 3.0.
- **Appendix A, "Migrating From Previous SunScreen Firewall Products,"** contains a table comparing the commands from SunScreen EFS, Release 2.0, and SunScreen SPF-200 to the equivalent commands used in SunScreen EFS 3.0.
- **Appendix B, "Command Line Reference**," documents the command-line interface.
- **Appendix C, "Services and State Engines**," lists the services and state engines supported by SunScreen EFS 3.0.
- **Appendix D, "Error Messages**," lists the error messages generated by SunScreen EFS 3.0.
- **Appendix E, "Glossary**," lists the terms and their definitions used in the SunScreen EFS 3.0 documentation.

Related Books and Publications

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

- *Applied Cryptography*
  Bruce Schneier
  John Wiley & Sons, 1996, 2nd edition, ISBN 0-471-12845-7
- *Building Internet Firewalls*
  D. Brent Chapman and Elizabeth D. Zwicky
  O'Reilly & Associates, 1995, ISBN 1-56592-124-0
- *Computer Security Policies and SunScreen Firewalls*
  Kathryn M. Walker and Linda Croswhite Cavanaugh
  Sun Microsystems Press, 1998, ISBN 0-13-096015-0
- *Firewalls and Internet Security*
  Bill Cheswick and Steve Bellovin
  Addison-Wesley, 1994, ISBN 0-201-63357-4
- *Handbook of Computer-Communications Standards*
  Volume 3: The TCP/IP Protocol Suite
  William Stallings, Macmillan, 1990
- *Internetworking with TCP/IP, Volume 1*
  Douglas E. Comer
  Prentice Hall, 1995, ISBN 0-13-216987-8
- *Network and Internetwork Security Principles and Practice*
  William Stallings
  Prentice Hall, 1995, ISBN 0-02-415483-0
- *Practical UNIX and Internet Security*
  Simson Garfinkel and Gene Spafford
  O'Reilly & Associates, 1996, 2nd edition, ISBN 1-56592-148-8
- *TCP/IP Illustrated, Volume 1 The Protocols*
  W. Richard Stevens
  Addison-Wesley, 1994, ISBN 0-201-63346-9
- *TCP/IP Network Administration*
  Craig Hunt
  O'Reilly & Associates, 1992
- *Network Security: Private Communication in a Public World*
  Charlie Kaufman, Radia Perlman, and Mike Speciner
  Prentice Hall, 1995
- *SKIP IP-Level Cryptography* [http://skip.incog.com/]
- Sun Software and Networking Security [http://www.sun.com/security/]

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunStore℠ Internet site at `http://sunstore.sun.com`.

Accessing Sun Documentation Online

The `docs.sun.com` 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/`.

Getting Support for SunScreen Products

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/contacting`.

What Typographic Changes Mean

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 `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root 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	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER 1

SunScreen EFS 3.0 Overview

This chapter discusses the following topics:

- What is SunScreen EFS 3.0, Revision B?
- SunScreen EFS 3.0, features
- Software and hardware requirements
- Compatibility with other SunScreen products
- Online help and documentation

What Is SunScreen EFS 3.0?

SunScreen EFS 3.0 is a versatile firewall used for access control, authentication, and network data encryption. SunScreen EFS 3.0 integrates the two SunScreen firewall products SunScreen EFS and SunScreen SPF-200. SunScreen EFS 3.0 consists of a rules-based, dynamic packet-filtering engine for network-access control, and an encryption and authentication engine that enables you to create secure Virtual Private Network (VPN) gateways by integrating public-key encryption technology. It is the first firewall to address high availability (HA) for standard based encryption. Secure administration is provided through an easy-to-use administration graphical user interface (GUI) through a Web browser.

Note – Stealth mode reflects the operation of the SunScreen SPF product; whereas, routing mode reflects prior releases of SunScreen EFS.

SunScreen EFS 3.0 consists of two components: *Screen* and *Administration Station*. The Screen is the firewall responsible for screening packets and for performing the necessary encryption and decryption. The Administration Station is where you define your security policy and from where you administer your Screen(s). The two components can be installed on separate machines for *remote administration* or on a single machine for *local administration*.

A machine can support as many as 15 interfaces, one of which should be the path to the external (public) network. Stealth and routing modes are supported on all SPARC and x86 systems listed on the Solaris supported hardware list (located at: `http://access1.sun.com/drivers/hcl/hcl.html`).

Caution – Manually deleting and then re-installing network interfaces also removes SunScreen EFS 3.0 from the interfaces.

Both stealth and routing modes support the following link adapters: SBus local (`le`, `be`) (10Mb/s) and Quad (`qe`) Ethernet; as well as X1059A SBus (`hme`), X1049A SBus Quad (`qfe`), X1032A 10/100 Mbps PCI (`hme`), X1033 PCI (`hme`), and X1034A PCI Quad (`qfe`) FastEthernet. Screens running in routing mode additionally support: FDDI, Token Ring, and ATM.

SunScreen EFS 3.0 uses open-standard SKIP (Simple Key-Management for Internet Protocols) technology, pioneered by Sun, for encryption, authentication, access control, and secure virtual private networks. SunScreen EFS 3.0 incorporates SunScreen SKIP 1.5 for Solaris. You must log into the Solaris command line to directly administer SKIP on the Screen.

Note – SunScreen SKIP new features includes support for 4096-bit Diffie-Hellman modulus and new DH primes.

See the *SunScreen SKIP 1.5 User's Guide* for further information regarding SKIP encryption and administration.

You can remotely administer SunScreen EFS 3.0 with any computer that has a supported version of SKIP and a Java browser compliant with JDK 1.1.3. SKIP software is available for Sun Solaris, Windows NT with Service Pack 3, Windows 95, and Windows 98 with PC SKIP patches.

Note – If the choice list flickers when using the HotJava browser, quit and restart the browser.

SunScreen EFS 3.0 Features

- **Centrally Managed Groups** – provides a way for you to manage multiple Screens with a set of common objects through a specific, primary Screen, as well as monitor logs on individual Screens in a centralized management group or HA cluster. The primary Screen, where the objects reside, can be managed by many different Administration Stations like in prior SunScreen firewall releases.
- **Stealth or Routing Modes** – allows you to designate interfaces in either stealth or routing mode on a port-by-port basis. Stealth mode, as a layered product, no longer boots from a CD-ROM nor requires an installation diskette, and operating system (OS) hardening is optional. High availability (HA) is accessible in both modes. Proxies work in routing mode only.
- **Data Organization** – increases the efficiency of data storage and retrieval by handling text data through a common access method. Common objects comprised of policy objects defining your security policy, are maintained by the `edit` sub-command of the new process `ssadm`, which is written in Java, that provides local as well as remote administration capabilities.
- **High Availability (HA)** – supports stealth and routing mode installations. The primary HA Screen manages secondary HA Screens in an HA cluster. A passive HA Screen within a HA cluster mirrors the state of the active Screen, which can be the primary or a secondary HA Screen. When the 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.
- **Logging** – allows you to search, sort, and filter log messages to find critical information quickly and easily. You specify the log size value and what information you want recorded in administrative log files when you set up SunScreen EFS 3.0. Once running, you can monitor logs using the browser and the command line in real time.
- **Network Address Translation (NAT)** – enables a Screen to map an internal network address to a different 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, and the state of the address map is monitored. You specify when a packet using ordered NAT translations is applied based on source or destination addresses.
- **Proxies** – provides content filtering and user authentication in routing mode only.
- **Time-of-day Rules** – allows you to define rules that are only active at specified hours.
- **Tunneling** – uses encrypted tunnels to hide network topology from intruders and to set up secure VPN gateways over insecure public networks.

Software and Hardware Requirements

TABLE 1-1 lists the installation requirements for SunScreen EFS 3.0.

TABLE 1-1 SunScreen EFS 3.0 Installation Requirements

Requirement	Description
Operating Environment	Solaris 2.6 or Solaris 7 in either 32-bit or 64-bit mode for SPARC and x86 systems.
Browser	Web browser compliant with JDK™, Release 1.1.3 or later. Browsers supported: • HotJava 1.1 on Solaris running on SPARC, x86, and Win95, Win NT 4.0, and Win98 • Netscape 4.5 and Internet Explorer 4.01, or higher, can be used to perform 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	All supported Solaris 2.6 or Solaris 7 for SPARC and x86 systems.
Disk Space	Minimum of 1 GB (at least 300Mbytes unused)
Memory	• For machines running just the Screen: 32MB Minimum • For machines running the Administration Station: 32MB Minimum, 64MB *strongly* recommended
Network Interface Supported	The Screen supports all Solaris supported Ethernet adapters. Additionally, when in routing mode, the Screen also supports: • Sun FDDI • Sun Token Ring (Screen only) • Sun ATM* (Screen only) The Administration Station supports all Solaris supported Ethernet adapters, plus Sun FDDI.
Link Support	Ethernet, Fast Ethernet, ATM (155 and 622 Mbit/sec in LAN Emulation mode), Token Ring, and FDDI. High availability and stealth-mode interfaces are *only* supported for Ethernet and Fast Ethernet. Also, high availability requires that the two boxes are connected in a non-switched hub.
Media	CD-ROM drive and diskette drive

* Currently, there is no support for Classical IP (cip) over ATM. There is support for lane (LAN emulation), only.

SunScreen EFS 3.0 includes the HotJava 1.1 software and the SunScreen SKIP for Solaris software.

Note – Due to a limitation in SunScreen SKIP 1.5 for Solaris, the RC2 encryption algorithm is not available when running Solaris 7 in 64-bit mode.

Required Patches

Two patches, which are included on the SunScreen EFS 3.0 CD-ROM, are required when you are running the Solaris 2.6 operating environment.

Apply the patches to a SPARC system by typing:

```
# cd /cdrom/cdrom0/sparc/Patches
# patchadd 106125-06
# patchadd 105181-11
```

Apply the patches to an x86 system by typing:

```
# cd /cdrom/cdrom0/i386/Patches
# patchadd 106126-06
# patchadd 105182-13
```

Java Plug-In Software

Java plug-in software system requirements:

Windows 95, Windows 98, or Windows NT 4.0

- Pentium 90 MHz or faster processor
- 10 MB free hard disk space (recommended 20 MB)
- 24 MB system RAM

Sun Solaris 2.5 or above

- Sun SPARC or Intel x86 microprocessor
- 10 MB free hard disk space (recommended 20 MB)
- 32 MB system RAM (recommended 48 MB)

Java Plug-in software, which is provided free-of-charge, is available at the following URL: `http://java.sun.com/products/plugin/1.1.2/index-1.1.2.html`

Java Plug-in software enables you to direct Java technology-enabled applets on your Intranet Web pages to run using Sun's Java Runtime Environment (JRE), instead of the browser's default runtime. They enable you to support Microsoft Windows- and Sun Solaris-based browsers in your enterprise.

See Appendix A, "Using the Command Line," in the *SunScreen EFS 3.0 Administration Guide* for instructions on how to install the plug-in software.

Compatibility With Other SunScreen Products

This release differs from previous releases of SunScreen firewall products both in how the Administration Station and Screen communicate with each other and in how rules are defined. You can use the SunScreen EFS 3.0 administration GUI (Web browser) software to manage SunScreen EFS 3.0 and SunScreen EFS, Release 2.0; however, you cannot use it on any other previous versions of SunScreen firewall products. The `ss_client` command is maintained so you can still remotely manage other Screens through the command line.

Note – See Appendix A, “Migrating From Previous SunScreen Firewall Products” for information regarding command compatibility with previous releases; and for information regarding the current commands for SunScreen EFS 3.0, see Appendix B, “Command Line Reference.”

The SunScreen SKIP encryption system built into SunScreen EFS 3.0 is completely compatible with other SKIP implementations, such as prior releases of SunScreen firewall products, *SunScreen SKIP for Solaris*, or *SunScreen SKIP for Microsoft Windows*. SunScreen EFS 3.0 can exchange encrypted information with other SunScreen firewall products transparently.

To upgrade to SunScreen EFS 3.0 from prior SunScreen firewall releases, see the upgrading instructions in your *SunScreen EFS 3.0 Installation Guide*.

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 on the SunScreen navigation buttons banner.

The SunScreen EFS 3.0 CD-ROM includes a documentation directory that contains files in Hypertext Markup Language (HTML) and Portable Document Format (PDF) format.

Click the Documentation button for the HTML files. They are located in: `/opt/SUNWicg/SunScreen/admin/htdocs/html`.

PDF files use the file extension identifier .pdf. To display or print a .pdf file, use the Adobe freeware Acrobat Reader. The .pdf files are located in:
`/opt/SUNWicg/SunScreen/admin/htdocs/pdf`.
They can be displayed using the command:
`imagetool /opt/SUNWicg/SunScreen/admin/htdocs/pdf/`*filename* &

Documentation is available on Sun's external document Web server `docs.sun.com` and stored in Sun's documentation archive. For a list of documents and how to order them, see the catalog section of the SunStore[SM] Internet site at
`http://sunstore.sun.com`.

The `man` pages for SunScreen EFS 3.0 are located in:
`/opt/SUNWicg/SunScreen/man`.

The `man` pages for SunScreen SKIP are located in the standard Solaris man page directory.

CHAPTER 2

SunScreen EFS 3.0 Concepts

This chapter describes the SunScreen EFS 3.0 concepts:

- Security considerations
- How SunScreen EFS 3.0 works
- Routing and Stealth modes
- SunScreen EFS 3.0 function details

Security Considerations

A company's assets are at risk when it connects to the Internet. It might want to provide Internet services for customers and other users of the Internet, while allowing its employees to connect to the Internet for services or access to corporate information.

SunScreen EFS 3.0 divides the world into discrete areas, each served by an interface. You set up filtering rules to control the 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 routing mode functions as a firewall and router to connect the Engineering network over an unsecured public network (the Internet) through a Screen in stealth mode to other secure networks.

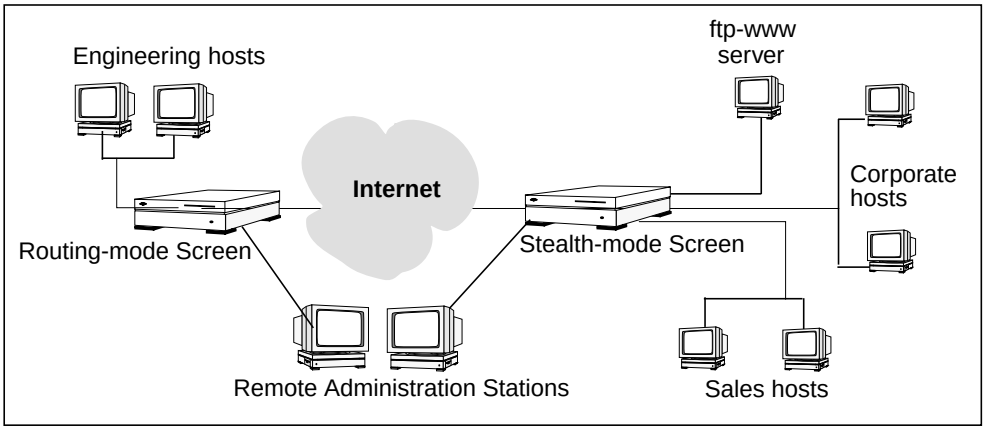


FIGURE 2-1 Sample Network Map

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 EFS 3.0 can then control access between these areas and the rest of the Internet.

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.

When defining your security policy, consider the following factors:

- To what services do employees need access?
- To what services or information do customers or other Internet users need access?
- Against what or whom are you trying to protect your company?

A security policy is a protective device; therefore, it is necessary to determine what you are trying to protect and from whom. Once you have identified your security requirements for protecting the integrity and accessibility of your corporate data and computer resources, determine what services you want to support at your site for employees and customers.

To help determine your requirements, use the following questions:

- Do users need to transfer files outside the organization?
- Will users be downloading files from sites outside your own network?
- What access to corporate data (that is, customer support or product information) do you provide customers?
- Who needs to log in remotely from other locations?
- Will you need to use "private" addresses so that you can:

- Support more Screens or subnetworks than are available from your Internet Service Provider (ISP)?
- More easily renumber your network and Screens, should you change ISPs?
- Use unregistered Internet addresses?

Once you have determined the answers to these and any other site-specific security issues, you are ready to plan your SunScreen EFS 3.0 configuration.

Policy rules are used to control access to your computer network and to control encryption for access to your data. By default, SunScreen EFS 3.0 drops any packets that do not specifically match a rule. This makes it easier to create rules, since you only have to write a rule for the services you want to pass.

To prepare to implement policy rules, you must:

- Identify the overall services available on your network
- Identify the services available to a particular user or Screen or to a group of users and their IP addresses

---

Note – Take care when defining Screen names. See "Define Screen's Name Properly."

- Determine the correct action for the services and addresses for each user or Screen

See the *SunScreen EFS 3.0 Administration Guide* for worksheets to assist you in gathering the information you need for setting up your security policy.

Define Screen's Name Properly

SunScreen EFS 3.0 automatically chooses a name for each Screen based on the hostname setting output by `uname -n`. There are various situations in which this name is used as an IP host name (IP address) for remote administration and centralized management groups.

Therefore, it is necessary for each Screen's name to be defined as a valid IP address for that Screen. The definition must be accessible through `/etc/hosts`, NIS or DNS on every remote administration station as well as every Screen in a centralized management group.

How SunScreen EFS 3.0 Works

SunScreen EFS 3.0 is a Solaris software product supporting Solaris 2.6 and Solaris 7 for both SPARC and X86 systems.

Note – Upgrade your system to at least Solaris 2.6, as SunScreen EFS 3.0 cannot support Solaris 2.5.1 due to Unicode internationalization requirements.

The administration GUI software works on any hardware or software system with a browser that supports JDK 1.1 (>= 1.1.3) and has End-system SKIP installed.

Integration of the two SunScreen firewall products in SunScreen EFS 3.0 allows you to create a stealth-mode firewall as a dedicated perimeter defense and extranet firewall, or a routing-mode firewall as a traditional firewall on the perimeter of a network or a remote-access server inside the intranet to segregate departments, or deployed on an existing application or data server throughout an enterprise to control access and provide encryption.

SunScreen EFS 3.0 and SunScreen SKIP use graphical user interfaces called:

- SunScreen EFS 3.0 installation wizard
- SunScreen EFS 3.0 administration GUI
- SunScreen SKIP `skiptool` GUI

See the *SunScreen SKIP 1.5 User's Guide* regarding the `skiptool` GUI.

The installation wizard allows you to configure your Screen in routing mode, which is the default, or in stealth mode. Following installation, use the administration GUI to administer your Screen locally on the same machine or remotely from an Administration Station. Use the `skiptool` GUI to encrypt administration commands that travel from the Administration Station over a potentially insecure network to the Screen.

Note – For backwards compatibility, installation for SunScreen EFS 3.0 retains the `ss_install` command.

The `ssadm` sub-command `edit`, which invokes the configuration editor, maintains the SunScreen EFS 3.0 configuration database.

A configuration is the union of one policy with the common objects to form a complete description of the behavior of one or more Screens. A policy is a named set of policy objects. For example, when the SunScreen EFS 3.0 software is first installed, there is one policy, named "Initial." Common objects are data objects relevant to all policies. Object types are either named or ordered. Named common object types

include Address, Screen, State Engine, Service, Interface, Certificate, and Time objects. Ordered policy object types include filtering rules, NAT rules, administration access rules, and VPN gateway descriptions. Neither common objects nor policy objects include objects loaded into SKIP but they do include the reference from the Certificate name in the common object registry to the internal identity used by SKIP.

The centralized management group feature allows Screens located at different points in the network to be managed with a standard set of objects through an Administration Station.

The high availability (HA) cluster feature allows groups of Screens to be deployed for failover protection. One member of the HA cluster, the active Screen, services packets travelling between a protected inside network and a nonsecure 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. 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.

The network address translation (NAT) feature enables a Screen to map an internal network address to a different 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, and the state of the address map is monitored. You specify when a packet using ordered NAT is applied based on source and destination addresses.

Individual versions of a policy are copied or saved into a new policy. Each version of a policy is maintained and you can use either all or a portion of a policy at a later date.

How SunScreen EFS 3.0 Uses Encryption

SunScreen EFS 3.0 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 devices can be encrypted, while all traffic between a Screen and an Administration Station is encrypted.

When the rules and policies determine that a specific packet should be encrypted, the new packet is first checked to see if it is too large to send on.

Encrypted packets are larger than the original packet for three reasons.

1. The original packet is encapsulated inside a new IP packet for transmission over the Internet.
2. A SKIP header is added so that the receiver can decrypt the packet.
3. 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 requesting 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. This allows the other end of the encryption tunnel to 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 tunnelling is specified, the tunnel source and destination addresses.

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

Note – Due to a limitation in SunScreen SKIP 1.5 for Solaris, the RC2 encryption algorithm is not available when running Solaris 7 in 64-bit mode.

When an encrypted packet is received it is passed to the decryptor. By examining the SKIP header, it 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. It then decrypts the traffic key and in turn decrypts the original IP packet.

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

SunScreen EFS 3.0 uses encryption in a feature called tunneling that is used to hide actual source and destination addresses. In this feature, you can substitute the addresses on the packet header with other addresses. When the SunScreen EFS 3.0 encrypts a packet, it replaces the packet's source address with the (optional) *tunnel address* of the *From Encryptor* and replaces the packet's destination address with the (optional) tunnel address of the *To Encryptor.* When the SunScreen EFS 3.0 encrypts a packet, the original addresses are restored.

Note – SunScreen EFS 3.0 incorporates cryptography at the network (IP) layer to provide privacy and authentication over unsecure public networks, such as the Internet. See the *SunScreen SKIP 1.5 User's Guide* for full descriptions of these and the Certification Authority (CA) issued keys and certificates.

Remote and Local Administration

SunScreen EFS 3.0 consists of two components: a *Screen* and an *Administration Station*. The two components can be installed on a Screen and a remote Administration Station, or they can be installed locally on a single machine.

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.

A machine that is being administered remotely can be *headless* (no monitor) and have no keyboard. You typically choose whether to administer a Screen locally or remotely when you install the SunScreen EFS 3.0 software. You can add a remote Administration Station after the Screen software has been installed.

Remote Administration

Remote administration from an Administration Station to the Screen, installs the software packages, including SunScreen SKIP, on separate machines, as shown in FIGURE 2-2. Because administration commands travel from the Administration Station to the Screen over a potentially insecure network, commands are encrypted using SunScreen SKIP.

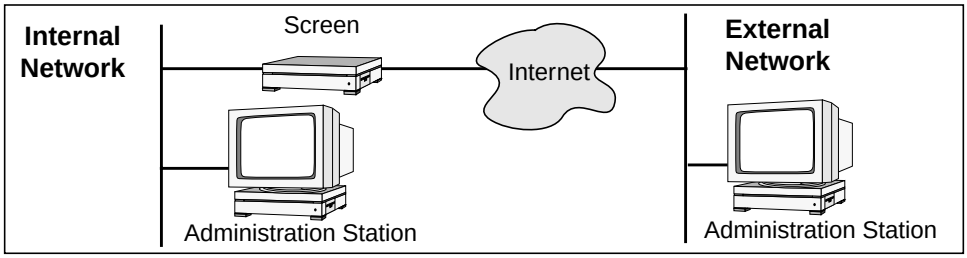


FIGURE 2-2 Remote Administration From an Administration Station to a Screen in Routing Mode

In FIGURE 2-2, 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. Either Administration Station can be configured to communicate with the Screen using encryption.

Local Administration

Local administration is performed on the same host where the Screen software is installed, as shown in FIGURE 2-3. Because administrative commands do not travel over a network, local administration does not require encrypted communication.

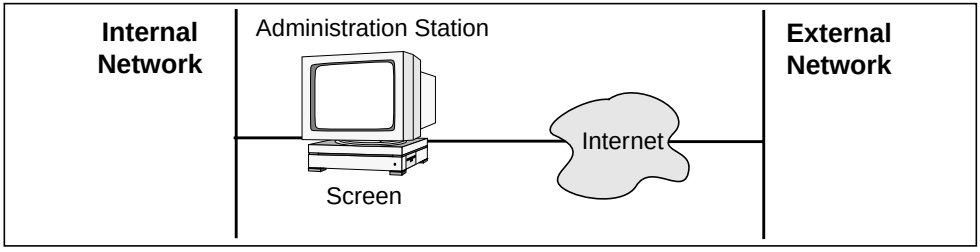


FIGURE 2-3 Local Administration of a Screen in Routing Mode

Centralized Management of Firewall Groups

Groups of Screens deployed throughout the world are managed with a set of configuration objects through an Administration Station. Policy objects reside on a specific Screen called the centralized management group's primary Screen. As in prior releases of SunScreen firewall product lines, Screens can be managed by many Administration Stations.

The centralized management group's primary Screen, where all configuration objects reside, manages the centralized management group's secondary Screens. centralized management group secondary Screens allow basic emergency administration capabilities; for example, if the primary Screen is down for service. Although there is no central logging mechanism for a global view of the logs on the individual Screens in a centralized management group, you can select a specific Screen and view its log.

Common objects are named objects, like address, screen, state engine, service, interface, certificate, and time. Policy objects are *ordered* filtering rules, NAT rules, administrative access rules, and VPN gateway descriptions. Neither common objects nor policy objects include objects loaded into SKIP but they do include the reference from the Certificate name in the common object registry to the internal identity used by SKIP.

Setting up rules for the entire centralized management group of Screens is done through the administration GUI.

Setting Up Access Control

Access control determines which users and Administration Stations can administer the Screen. You choose local or remote administration by selecting the Administrative Access tab in the Policy Rules area of the administration GUI. The "Access Rules for GUI Local Administration" table under local administration lists rule No, Screen, User, Access Level, and Description.

You can use the configuration editor's syntax through the command-line interface (see Appendix B for command line reference) to specify which machines on your network can administer a Screen as well. The configuration editor is the primary command-line tool for creating and manipulating the data objects that control the operation of a Screen.

Policy Versions

Each *version* of a policy has an associated version number. Edited policies automatically generate historical version numbers. Version numbering is implemented through the administration GUI using the `edit` sub-command of `ssadm`, which allows you to create new policy object files by name.

When a particular policy is superseded by a new version of that policy, the older version includes the policy object-specific information and the actual content of the common object registry instead of just a reference. Although you may find that the older policy version has become invalid due to changes that occurred in the network and hardware topology, it can be more consistent.

Routing and Stealth Modes

SunScreen EFS 3.0 includes stealth-mode capabilities and routing-mode capabilities.

Routing Mode

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

All proxies are accessed through the transmission control protocol (TCP), and therefore can only run on systems configured in routing mode.

Stealth Mode

Stealth-mode offers optional hardening of the OS, which removes packages and files from the Solaris operating system that are not used by SunScreen EFS 3.0. Stealth-mode requires the Screen to partition a single network.

Stealth mode firewall partitions an existing network and, consequently, does not require you to sub-net the network. Stealth-mode interfaces do not have IP addresses, and do MAC-layer bridging.

In stealth mode, SunScreen EFS 3.0 is similar to the SunScreen SPF-200 product; however, it differs from it in the following ways:

- It is a layered product instead of a dedicated installation.
- It no longer boots from the CD-ROM.
- It no longer requires an installation diskette.
- (Optional) Hardening of the operating system (OS) is equal to SunScreen SPF-200 when the minimum required OS packages are installed with the minimum required patches.

If you configure a network interface that you later set to Stealth mode, the Screen will hang upon activation. If this happens, you must reboot the Screen in single-user mode, then remove the `/etc/hostname.`*interface_name* file (which unconfigures that interface), and reboot the Screen again.

If you accidentally misconfigure the system in this way, here is the procedure for restoring proper operation:

1. **Type `control-C` a few times to break out and send your machine into single-user mode.**

2. **After typing your root password, you must type the following to remount the root partition read-write:**

```
# mount -o remount /
# ls /etc/hostname*
/etc/hostname.hme0   /etc/hostname.qfe2
```

The `qfe2` interface is the admin interface in this example.

Note – Do not disturb your admin interface, as it must be the only hostname interface file in the `/etc` directory.

As the example shows, the problem is the existence of a `hme0` interface file.

3. **To rename or remove the problem hostname interface file, type:**

```
# mv /etc/hostname.hme0 /etc/hostname.hme0.old
```

4. **Reboot the machine.**

```
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 270MHz), No Keyboard
OpenBoot 3.11, 128 MB memory installed, Serial #10411258.
Ethernet address 8:0:20:9e:dc:fa, Host ID: 809edcfa.
Rebooting with command: boot
Boot device: disk:a  File and args: kadb
kadb: kernel/sparcv9/unix
Size: 314284+93248+121472 Bytes
/platform/sun4u/kernel/sparcv9/unix loaded - 0xca000 bytes used
SunOS Release 5.7 Version Generic 64-bit [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1998, Sun Microsystems, Inc.
plumbing SunScreen network interfaces:^C
INIT: Cannot create /var/adm/utmp or /var/adm/utmpx
INIT: failed write of utmpx entry:"  "
INIT: failed write of utmpx entry:"  "
INIT: SINGLE USER MODE
Type Ctrl-d to proceed with normal startup, (or give root password for
system maintenance):
Entering System Maintenance Mode
May  5 17:10:04 su: 'su root' succeeded for root on /dev/syscon
Sun Microsystems Inc.    SunOS 5.7        Generic October 1998
# mount -o remount /
# ls /etc/hostname*
/etc/hostname.hme0   /etc/hostname.qfe2
# mv /etc/hostname.hme0 /etc/hostname.hme0.old
# ls /etc/hostname*
/etc/hostname.hme0.old   /etc/hostname.qfe2
# reboot
```

SunScreen EFS 3.0 Functions

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, as shown in the following figure.

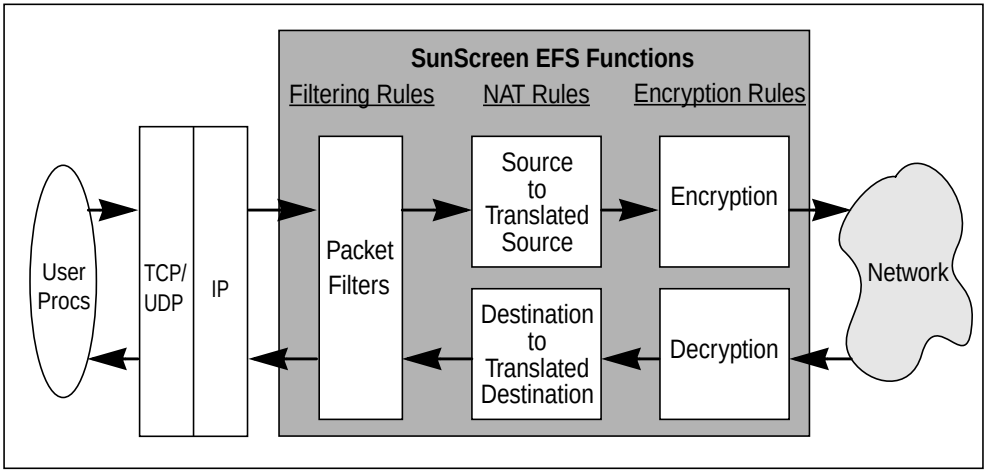


FIGURE 2-4 Rule Processing Order in a Screen

When a host on a private (internal) network sends a packet addressed to an outside host through a Screen, the Screen does the following:

1. The Screen applies its packet filter 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 a DENY rule, then the packet is denied and, if so specified, can make a log entry identifying the circumstances.

2. If the packet meets the criteria of an ALLOW rule, the Screen determines whether it should use NAT to convert the source IP address information in the packet. If NAT is being used, the source address information in the packet is modified. Depending on the type of packet, the Screen may also modify address information in the data portion of the packet or modify packet checksums.

3. The Screen determines if 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, which identifies the appropriate encryption keys and algorithms and encrypts the packet. The SKIP key manager passes the encrypted packet back to the Screen, which then forwards the packet to the public network.

When a Screen receives a packet addressed to an internal host from a host on the public network, it performs a similar sequence of actions:

1. If the incoming packet has been encrypted with the Screen's public key, the Screen calls the SKIP key manager, which identifies the appropriate keys and decrypts the packet. The SKIP key manager passes the decrypted packet back to the Screen.
2. The Screen determines whether it should use NAT to convert the packet's destination IP address to an internal IP address. If NAT is being used, the address information in the packet is modified.
3. 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 to the internal network. If the packet meets the criteria of a DENY rule, then the packet is denied.

SunScreen EFS 3.0 Function Details

The following SunScreen EFS 3.0 functions are described in more detail:

- Dynamic packet filtering
- Network address translation (NAT)
- High availability
- Time-based rules
- User authentication
- Event logging with proxies
- Encryption and decryption
- Address management
- Services and service groups
- Policy rules
- Tunneling and VPNs

Dynamic Packet Filtering

Dynamic packet filtering enables a Screen, which sits between the client and server, 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 EFS 3.0 uses a set of ordered rules to filter packets. When you configure SunScreen EFS 3.0, you translate the security policies for your site into a series of policy rules that specify which services are to be 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.

When the Screen receives a packet, it tests the packet against the two certificates, the algorithms, and the rules in the order in which they are numbered. The Screen does not test each packet against each rule; it assumes that the first rule to match the service, source address, and destination address of the packet addresses is the rule that governs the disposition of the packet. Additionally, if the applicable rule so specifies, the Screen can record a log message or generate an SNMP alert.

If the packet does not match any rule, the Screen uses its default action for the interface on which the packet arrived to determine disposition of the packet. Typically, this action logs the packet and drops it, though other options are available.

Network Address Translation (NAT)

SunScreen EFS 3.0 NAT gives you fine-grain control by adding ordered NAT translations, allowing table entries to intersect, and allowing you to specify when to have NAT translate the source or destination addresses.

NAT enables a Screen to map an internal network address to a different 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 map is monitored. You specify when a packet using ordered NAT translations is applied based on source and destination addresses.

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

Additionally, 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 in the following situations when:

- Renumbering all the hosts in your network is not feasible.
- The current private network uses a set of private, unregistered IP addresses due 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 Internet Service Provider (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 at each end of the connection over the public network.

NAT lets you use private or 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 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).

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, so the reverse rules must be generated by you.

Static NAT maps a specific unregistered address to a specific registered address. Static translations can also map a range of unregistered addresses to a range of registered addresses, which requires the number of addresses in each range to match.

Dynamic NAT

Dynamic NAT maps a large set of unregistered IP addresses to a smaller set of registered addresses. Dynamic NAT lets you connect a very 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 mapping between internal private addresses and external public addresses, dynamic NAT creates a many-to-one mapping 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. When the Screen uses a public address that is already in use, it uses a different source port number, thereby making a distinction between the two connections. A Screen can multiplex many thousands of translations over a single registered address by ensuring the source ports for the connections differ.

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.

NAT Examples

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

Example One:

When you only have one registered IP address (“A”) and you want to have all inbound traffic to “A” go to your Screen and have all other hosts use that address (“A”) for unidirectional, outbound traffic, then set up the following NAT mappings:

TABLE 2-1 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”	“”

where “Internet” is all addresses on inbound interface “`qe0`” and “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.

Valid mapping combinations are:

- “Source” and “Translated Source” are the same, and “Destination” and “Translated Destination” are the same (no translation will occur).
- “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 mapping or in a combination of mappings.

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, either machines that host public resources must have stable registered addresses, or their internal (unregistered) 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 FIGURE 2-5, 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.

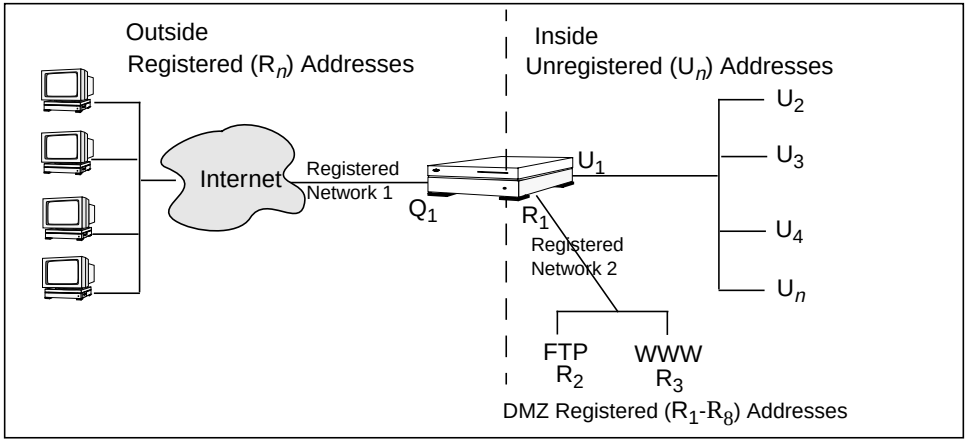


FIGURE 2-5 Scenario 1: Static and Dynamic NAT

The Screen uses dynamic NAT to map the addresses in its unregistered address range (U2–Un) to map 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.

Scenario 2: DMZ Uses NAT Addresses

FIGURE 2-6 illustrates an organization that has a network consisting of a large number of unregistered addresses (Un) and a set of eight registered addresses (R1–R8). Hosts on the inside network must be able to communicate through the Screen with external hosts.

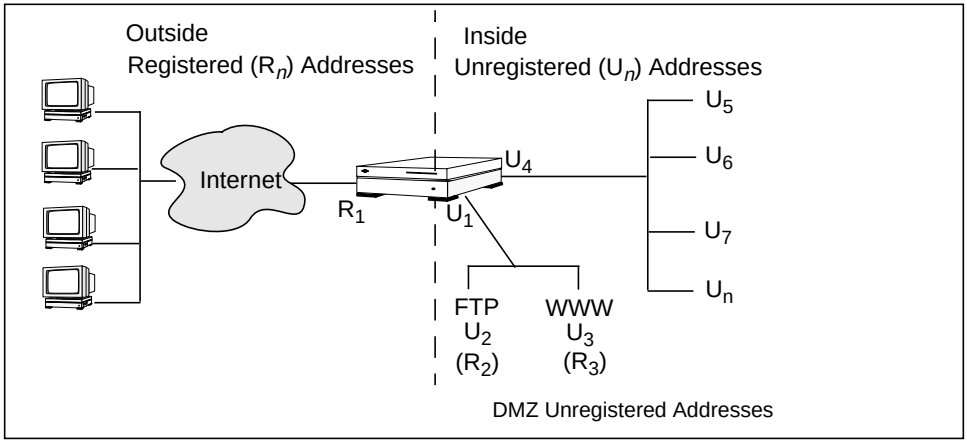


FIGURE 2-6 Scenario 2: Static and Dynamic NAT

In FIGURE 2-6, 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 map 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 Un will be mapped 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.

In routing-mode (not needed in stealth-mode), the Screen must respond to ARP requests the public addresses (R2 through R8) because it will be translating public addresses to private addresses. You must add an `arp` entry (using the command `arp -s` *IP_address ether_address* pub) for them. You must either add this entry each time that you reboot the Screen or add it to the `/etc/inetd.conf` file. If you are remotely administering the Screen in routing mode, you either must go to the Screen to add this entry, or you must have a rule in your policy that allows you to log in remotely (`rlogin`) to your Screen.

The Screen uses dynamic NAT to map the addresses in its unregistered address range (U4–Un) 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 readdress all the hosts on your internal registered network.

Applying NAT

NAT mappings 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.

NAT and Mapping Collisions

Mapping collisions can cause service to be denied to a network user. Mapping collisions occur when network software cannot complete the address mapping 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: (`dest IP addr`, `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. For a mapping collision to occur, multiple internal machines using the same registered IP address must try to access the same registered address Xn at the same destination port number, and from the same source port number, all at the same time-an unlikely scenario.

Suppose a user at the unregistered address U5, shown in FIGURE 2-5, 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 following table shows a mapping of unregistered addresses, Un, to registered addresses, Rn.

TABLE 2-2 Two Dynamic Addresses

Registered IP Address	Destination IP Address	Destination Port	Source Port	Protocol
R_4	192.4.15.37	80	34080 (on U_5)	tcp
R_4	192.4.15.37	80	34070 (on U_6)	tcp
R_4	192.4.15.44	80	34080 (on U_7)	tcp
R_5	192.4.15.44	80	34080 (on U_7)	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 mapping collision will occur. The user at U7 would have to use another source port to have a unique 5-tuple and avoid a mapping collision, which would happen automatically during a subsequent connection attempt.

Situations such as power failures typically result in mapping 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. It may take time for the counters on each machine to cycle up to higher and more randomized port numbers (which are more likely to produce unique 5-tuples). In the interim, mapping collisions may cause network service to be denied temporarily. Internal hosts must continue trying to establish network connections until the NAT rules resolve the mapping collisions.

Note – Ports 0 though 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

High Availability (HA)

HA lets you deploy groups of Screens together in situations where the connection between a protected inside network and a nonsecure 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 HA cluster, the *passive* 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.

HA Policy

When you set up an HA cluster, you designate one Screen as its *primary* HA Screen that is configured with the policy's configuration objects, including named Screen objects, like Address or Service with attributes that include these settings, and policy rules that the HA cluster will use. When you activate the security policy, the SunScreen EFS 3.0 and SunScreen SKIP policies are copied from the primary HA Screen to the *secondary* Screens in the HA cluster.

Solaris policy settings, such as network interfaces and routing configuration, are not copied from the primary Screen and must be identical on all the Screens in the HA cluster.

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

The interfaces for network connections must be the same for each HA cluster member. Similarly, you must assign all HA Screens the same IP addresses on their non-dedicated interfaces as well. The following figure shows a network protected by two Screens in an HA cluster. Each Screen in the HA cluster connects to the external and internal networks through Ethernet hubs, which pass the same signals to all HA cluster members at the same time. Each HA Screen therefore sees the same traffic, ensuring that passive Screens can duplicate the state of the packet filter engine should the active Screen fail.

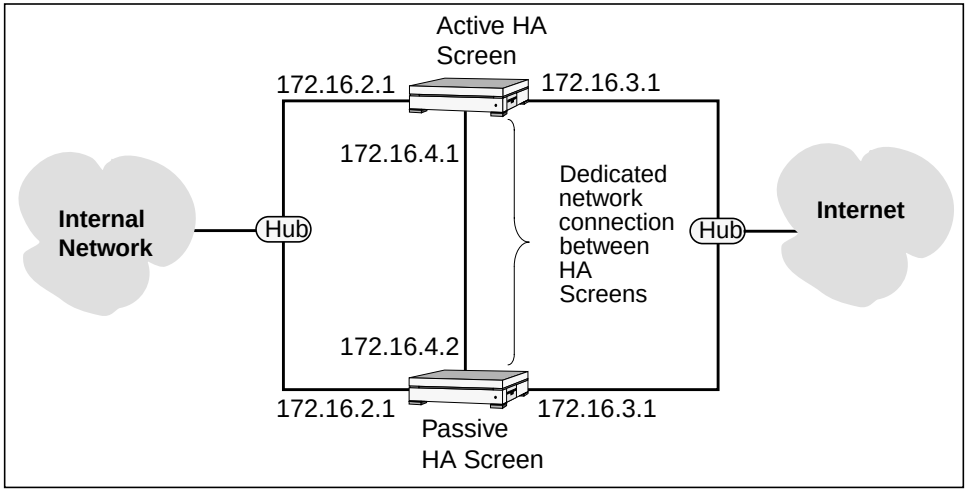


FIGURE 2-7 Network With HA Cluster of Screens

Note – When the active HA cluster Screen is an x86 machine running Solaris 7, failover does not work properly. The ETHER_ADDRESS for the primary does not set correctly.

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 as the active Screen within 15 seconds. During this time (before the passive Screen takes over), no traffic will go through the HA policy.

HA is designed to maintain the great majority of network connections. In the case of 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 affect the passive Screens.

If a failover occurs, these connections may be disrupted:

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

These connections may be lost if one of the following conditions occurs:

- The Screen taking over filtering does not have all the same state information when the failover condition occurs.
- Although rare, 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 machine, allowing it to act like a standard SunScreen EFS 3.0 Screen.

You choose to configure SunScreen EFS 3.0 as an HA cluster during installation. Alternatively, you can configure HA settings through the command line, as described in Appendix B, “Command Line Reference.”

HA Alerts

Note the following HA limitations:

- HA is currently only supported in routing mode.
- HA does not work on x86 machines running Solaris 7.
- Screens in an HA cluster do not exchange state information, they independently accumulate state by processing the same packets. For optimal performance, whenever possible, it is recommended that an HA cluster 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.
- To reinstate a repaired Screen back into an HA cluster, install it as a secondary Screen and add it to the cluster.

  It is not necessary to switch back to the original Screen except when its abilities or speeds are better. If they are, you can force the faster machine to take over as the active Screen, or wait until both machines have the same state because eventually the out-of-sync connections will timeout and become synchronized, then force the failover.
- Secrets are sent in the clear on the dedicated HA network. This is done intentionally to facilitate administrative communication within the HA cluster. Reserve the HA interface as a separate network from other traffic to avoid the possibility of the Screen's private certificates being discovered.
- Each Screen in an HA cluster must have a unique nodename 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).
- HA does not work well with proxies because proxy state tables are not present on all HA machines.
- The 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 it would send each packet to only one HA Screen, rather than to all of them).
- HA cannot be used on networks that use media other than Ethernet, such as FDDI, Token-Ring, or ATM.
- Because passive HA hosts do not generate traffic (other than automatic administrative traffic, such as HA administration and HA heartbeat), you cannot telnet into passive HA Screens. You can telnet into an active HA Screen only.
- In general, the HA cluster decides internally which member will be the active Screen. If you need to remotely administer the HA cluster, you may first need to connect to the secondary Screen and direct it to become passive so that you can communicate with the primary Screen.

Time-Based Rules

Time-based rules are time objects that are specified using a 24-hour clock in 5-minute increments. They allow you to set a policy's time-of-day, day-of-week, and so forth, rules. For instance, you may want to 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 like in prior SunScreen firewall releases. You can set a few time-based policy rules that reference the same set of hours, or you can specify specific hours covered in a specific day 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.

For example, Los Angles (LA) and New York (NY) have a three hour difference. If you only want the two sites to communicate when they are both within "regular" hours (that is, 8am to 5pm), then NY is available to communicate to LA between 11am and 5pm, and LA is available to communicate to NY between 8am and 2pm.

A down-side to this is that during hours that do not overlap, one of the two Screens allows traffic through while the other will 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
```

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

User Authentication

SunScreen EFS 3.0 allows you to configure user entities to authenticate individual administrators and to allow access through the firewall when using proxied services.

Authentication allows 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 5, “Proxies,” for more information regarding user authentication.)

SunScreen EFS 3.0 provides two distinct levels of user identification: Authorized User, through the `authuser` database, and Proxy User, through the `proxyuser` database.

Authorized Users

Authorized User is a named common object that describes an individual administrative 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` sub-command of the `ssadm` command line facility reference an Authorized User object.

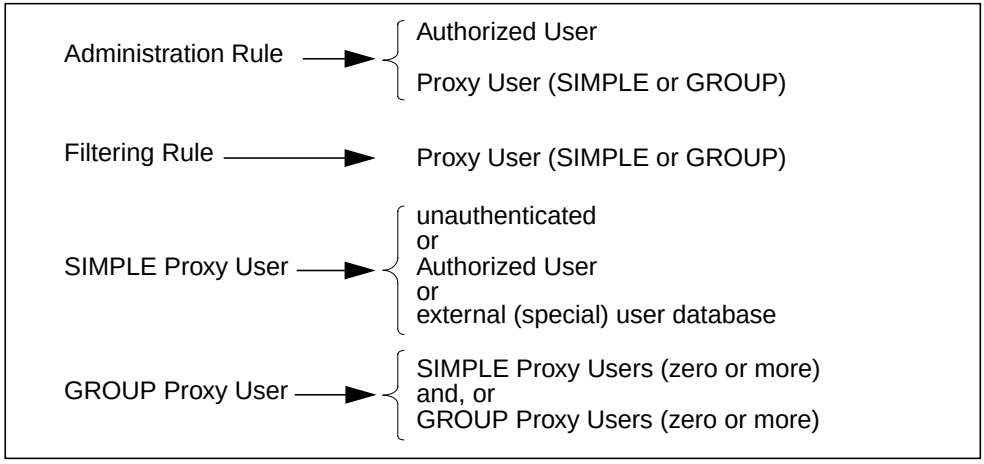
Note – Authorized User authenticity establishes only the identity of an individual administrator, not the various roles they may play while using SunScreen EFS 3.0. Role establishment is afforded in one of two ways: (1) reference within the User field in the administrative access rules of a policy, (2) reference from a packet filtering rule that utilizes user authentication (proxies).

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 an individual administrator and the role they play in usage of the facilities controlled by SunScreen EFS 3.0. 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 to established facilities.

Some special Proxy User objects also provide the means to *map* external collections of users into the SunScreen EFS 3.0 access control facilities. SunScreen EFS 3.0 provides external access to SecurID and RADIUS users. (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 following diagram summarizes 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 distinguished 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).

Note – Names cannot contain any of the following characters: "!", "#", "$", "%", "^", "&", "*", "{", "}", "[", "]", "<", ">", """, "'", "?", "‘", "/", "@", or NUL characters.

Space, tab, and other whitespace characters are allowed in names, but in doing so you should be prepared to supply quotation marks in some situations in order to protect such whitespace within names.

Names of Authorized Users, Proxy Users, and other user naming items are often deliberately chosen to be different for purposes of clarity.

Event Logging With Proxies

Event logging (failure and success, including subsystem authentication) through the log browser supports filtering and ordering capabilities to define and store named filtering and sorting *macros*. The size of the log is configurable. The log collection facility allows the proxies to add information to the current log. The types of logged events are extensible to anticipate the evolving use of this facility. Filtering and ordering automates uploading, storage, and post-processing of logs. You can create post-processing of your choice of uploaded logs (for instance, analysis and compression).

Encryption and Decryption

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. The original message is called the *plaintext message*. The encrypted message is called the *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 EFS 3.0 because it provides a mechanism for protecting the privacy of communications and authenticating the identities of the sender and receiver. Without encryption, you would have to define packet screen rules broadly: “all the machines on the Internet” and “all the machines on the inside.” Encryption technology lets you authenticate machines and users. As a result, you can define rules that control access by specific cryptographic identities rather than by general IP addresses.

SunScreen EFS 3.0 uses the SunScreen Simple Key-Management for Internet Protocols (SKIP) as the basis for its encryption technology. SKIP provides secure, encrypted communication between a remote Administration Station and the Screen and between a Screen and a remote SKIP host.

For detailed information on how SKIP encryption works, refer to the *SunScreen SKIP 1.5 User's Guide*.

Address Management

SunScreen EFS 3.0 identifies network elements—networks, subnetworks, and individual hosts—by mapping a named *address object* to one or more IP addresses. SunScreen EFS 3.0 uses address objects to define the network elements that make up the policy. These address objects are then used in defining SunScreen EFS 3.0's network interfaces and as the source and destination addresses for rules and for NAT. An address object can represent a single computer or a whole network. You can gather address objects representing individual and network addresses together to form address groups. SunScreen EFS 3.0 lets you define address objects that specifically include or exclude other address objects (single IP hosts, ranges of contiguous IP addresses, or groups of discontiguous IP addresses).

Individual IP Addresses

SunScreen EFS 3.0 identifies an individual host by linking its unique IP address to an address object, which can use the name or IP address of the host or some other identifier. FIGURE 2-8 shows an example of a host identified by its IP address (172.16.1.1).

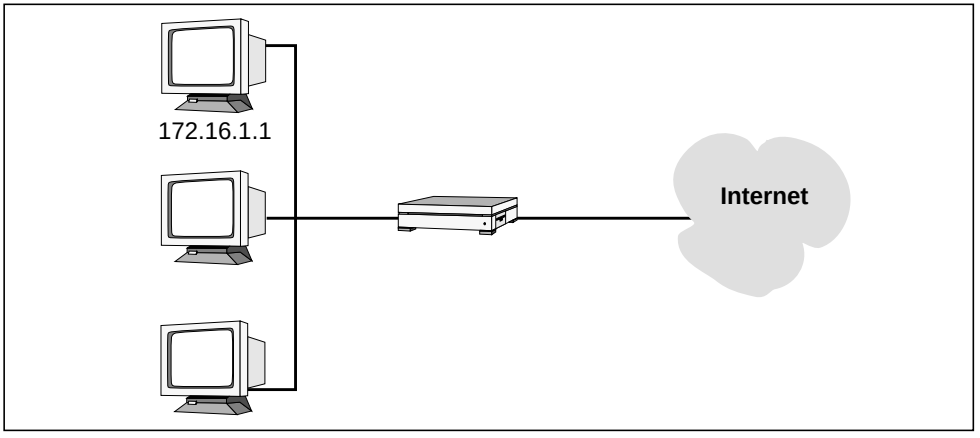


FIGURE 2-8 Single Host Address

Address Ranges

An address range is a set of numerically contiguous IP addresses. Networks and subnetworks are typically identified by an IP address range name. 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 EFS 3.0. FIGURE 2-9 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.

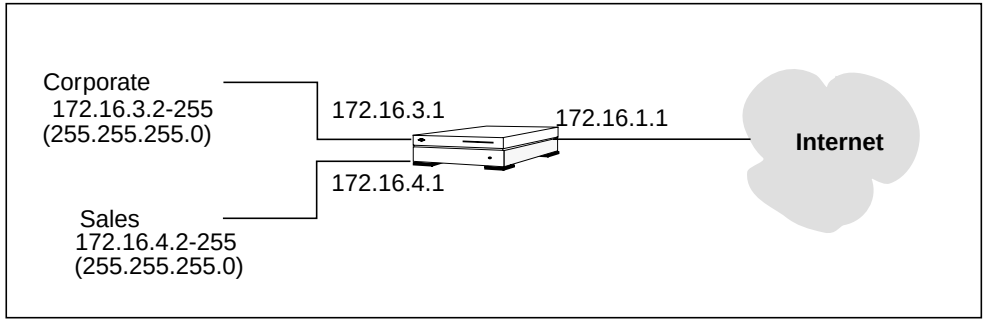


FIGURE 2-9 Examples of IP Address Ranges

Address Groups

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(es) of the actual SunScreen, and “*”, which represents all IP addresses.

The value is determined by first, all Addresses are included, which means that all the IP addresses contained in any of them are added to the Address Group. Next, all IP addresses of all the Addresses that are excluded are removed from the Address group. You cannot control the order in which the IP addresses are added or removed, as 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 EFS 3.0 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 usually has the majority of addresses on it. For example, the Internet network interface (`qe0`) in the network illustrated in the figure, has the most addresses, since it is the interface for all addresses except those in the `Corp`, `ftp-www`, and `Sales` networks.

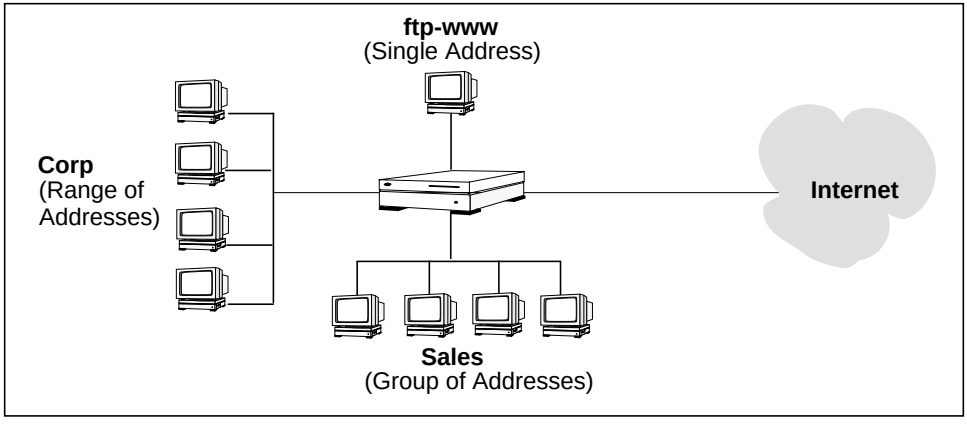


FIGURE 2-10 SunScreen EFS 3.0 as Internet Firewall

Rather than enumerating all the addresses for the Internet, you can define the address group for the Internet address object to includes 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.

Services and Service Groups

SunScreen EFS 3.0 is shipped with a number of predefined network *services*, such as `ftp`, `telnet`, `dns`, and `rsh`. 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.

Standard Services

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, most sites 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 may want to allow anyone in the internal corporate network to send outbound traffic, but only allow inbound traffic in this protocol to go to the 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. For example, the FTP state engine checks port numbers when the `ftp` service is being used. For more information on state engines, see Appendix B, “Services and State Engines.”

Table C-1 in Appendix C, “Services and State Engines,” lists the single services in SunScreen EFS 3.0, along with the state engine and discriminator (port, RPC program number, or type). Parameters (state engine modifiers, such as timeouts) and BROADCAST are indicated where applicable.

Modifying or Creating New Services

You can modify or define existing services or define new services as needed. Each service must use a predefined state engine. 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.

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

- What protocol does the new service use?
- What ports 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 B, "Services and State Engines," for more information on state engines, their discriminators, and their parameters.

You can specify state engines as filters for both the forward and the reverse direction. The forward filters apply when traffic originates from the From Address and goes to the To Address in a rule. The reverse filters apply 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 always gets 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 nonstateful traffic to return. An example is the `nlm` rule, which uses the nonstateful ICMP filter engine. It allows network lock manager (`nlm`) requests (ICMP type 8) in the forward direction and `nlm` replies (ICMP type 0) in the reverse direction.

State engines' discriminators can optionally be tagged with a BROADCAST attribute. 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 line both with and without BROADCAST selected.

Service Groups

You can group network services together to apply a single rule to multiple network services. This group is called a *Service Group*. Table C-1 in Appendix C, "Services and State Engines," shows the predefined Service Groups in SunScreen EFS 3.0 and the services each includes. Not every service is included in a service group.

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.

Policy Rules

Policy rules are based on sets of *ordered policy rules*. You set up these 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.

When you install the SunScreen EFS 3.0 software, you start with a policy object 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 if 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 Encryption or Allow action.

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

You can use the administration GUI or the command line (see Appendix B) to set up the table of ordered rules for a policy, which allows you to edit rule components, such as source and destination addresses, directly. You can change the order of the rules by dragging and dropping them through the configuration editor.

Rule Syntax

The basic syntax for a rule is:

```
Service Source_address Destination_address optional_Time
optional_Encryption "action" optional_Proxy
```

where:

- Service – An individual network service or a group of services provided to users. Sample services are email, remote login, and ftp.

- Addresses – Named addresses to define the network elements that make up the configuration. A named address can represent an individual address, as a range of addresses, or as a group of addresses. Named addresses are also used to define the network interfaces.
  - Source_Address – An address object from the client side over which these services are provided to users.
  - Destination_Address – An address object from the server side over which these services are provided to users.
- Encryption – (Optional) The version of SKIP to used for encryption, in one of two forms: SKIP_VERSION_1 *source_cert dest_cert key_algorithm data_algorithm*, or SKIP_VERSION_2 *source_cert dest_cert key_algorithm traffic_algorithm mac_algorithm*.

Note – Encryption is only supported if ALLOW 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 on the Screen), then 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, then 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 or if both certificates are local, then the rule is ignored by the compilation process.

- Action – What the Screen should do with packets. The choices are ALLOW and DENY. The syntax for specifying an action is in the form: ALLOW LOG_information SNMP_information; for example, ALLOW LOG_SUMMARY SNMP_NONE or DENY LOG_INFORMATION SNMP_INFORMATION ICMP_INFORMATION.
- 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. The syntax for specifying a proxy is in the form: PROXY_*proxy_name proxy_name flags*; for example, PROXY_HTTP HTTP *flags*, PROXY_FTP FTP flags, or PROXY_SNMP SNMP *flags*. The telnet proxy does not take flags.
- User – (Optional) A name in the proxy database of users. The 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 *only* supported if the rule is Proxy-ftp or Proxy-telnet. Otherwise, the Screen does not support user-based rule criteria.

- Time – (Optional) A component that allows time-of-day-based Policy Rules. The syntax for specifying time is in the form: *time_object_name*.
- VPN – (Optional) A component that allows you to no longer specify all the SKIP information, but have the Screen do it for them. The syntax for specifying VPN is in the form: *vpn_name*.

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

Example of a Rule Configuration

The XYZ Company wants to set up a series of rules to implement the following security policies:

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

TABLE 2-2 illustrates the rules the XYZ Company would set up to implement this security policy. Note that the default action would be specified as DENY for each interface to implement policy 4.

TABLE 2-3 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

Tunneling and VPNs

Organizations typically have offices in more than one location. SunScreen EFS 3.0 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.

Note – When editing multiple rules with ENCRYPT as the ACTION value, the tunnel address settings of the first rule you edit is copied into the next rule you edit. Use the command-line interface to modify the tunnel address settings of these rules.

When a tunnel, or *virtual private network*, 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 will 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 decapsulated, 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.

Note – When using encryption with tunneling in 64-bit mode, the destination address is incorrectly translated. 32 -bit machines are unaffected. To set a 64-bit installed machine to boot in 32-bit mode, go into firmware (the “ok” prompt) and set the boot file to `kernel/unix`, by typing: **`ok setenv boot-file kernel/unix`**.

FIGURE 2-11 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.

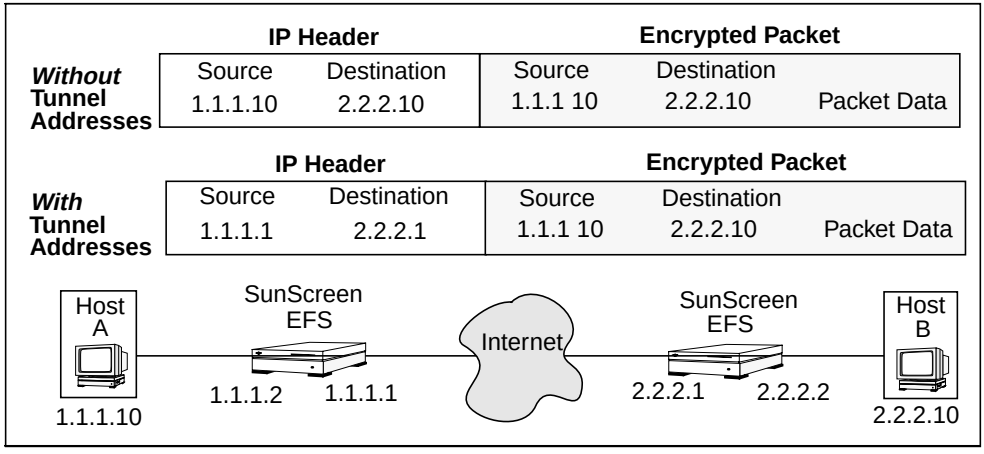


FIGURE 2-11 Effect of Tunneling on Contents of IP Headers

If you do not use a virtual private network (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 would not be able to read its contents but could 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 virtual private network 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 nonsecure 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.

Note that, since 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 2-12.

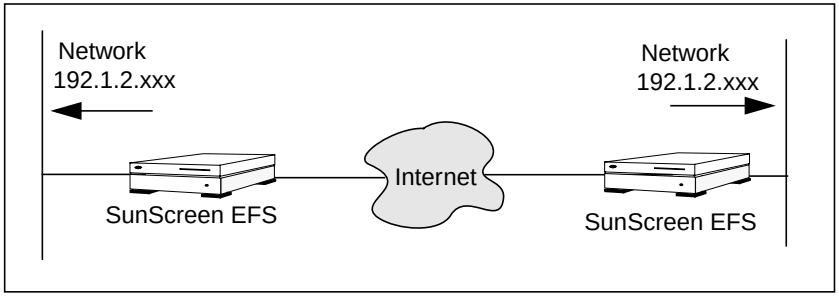


FIGURE 2-12 Geographically Dispersed Network

CHAPTER 3

Administration Graphical User Interface Reference

This chapter describes the administration GUI that you use to configure SunScreen EFS 3.0. It discusses the following topics:

- Starting the administration GUI
- Elements and navigation of the administration GUI

Starting the Administration GUI

The SunScreen EFS 3.0 administration GUI uses Java applets to administer and monitor Screens. When administering a Screen remotely, you start the administration GUI by opening a Java-aware browser, such as HotJava or Netscape Navigator/Communicator, and pointing it to `http://`*screenname*`:3852`. If you are administering a Screen locally, type `http://localhost:3852` in the browser. The Welcome page appears when your browser connects to the designated Screen.

Note – If your network uses proxies (gateways) for Internet access through a network firewall, you may need to configure your browser to ignore proxies when you are connecting to a Screen. For example, you might type `localhost` in the Don't Proxy field for your browser to specify that your browser should connect to `localhost` directly. For information on configuring proxies, refer to your browser documentation.

Tip – You can improve the performance of the administration GUI by clearing cached images out of your browser before you load the login page.

Administration Limitations

The command line offers many options for each command. The administration graphical user interface (GUI) performs almost all the normal administration tasks, but the command line is more powerful. The administration GUI does not support every option of the commands.

Error messages from the administration GUI are as informative as those from the command line. If an error message appears on the Java console of the browser, start the Java console and then try the action again to see the error.

Note – If the following error message appears on the console of the Screen after a reboot when running the administration GUI, “Error opening access control file,” no user action is required. It does not in any way affect the security of the system.

Elements of the Administration GUI

The administration GUI for SunScreen EFS 3.0 is organized as a set of *pages*. Each page consists of one or more controls. For example, most pages include buttons and text fields, which you update using your keyboard and workstation mouse.

Note – All information regarding the command-line interface is in Appendix B of this manual.

Welcome Page

The SunScreen EFS 3.0 Welcome page opens when you type `http://localhost:3852` , `http://`*ScreenName*`:3852` in a supported browser.

The Welcome page logs you into the administration GUI. The default user name and password is `admin` and `admin`, respectively.

Caution – Delete or change the password for the default login account as soon as possible to prevent unauthorized access to the Screen’s policies.

For a description on how to change passwords, see the *SunScreen EFS 3.0 Administration Guide*.

Once you are logged in, the first page shows buttons on the top banner: Logout, Policies, Information, and Documentation; two instructions follow the top banner that say: “To edit a policy select one from the table and click the “Edit” button,” and “For other tasks select from the top area buttons”; a banner follows titled “Policies List”; and below the Policies List banner is a table.

The table shows the Name of your policy, Version number, if present, and Active Policy Information, if present.

Beneath the table you have a choice of buttons: Add New..., Edit, Copy..., Rename..., Delete..., Activate, Backup All..., Restore All..., Initialize HA..., and Help...

- Click the buttons on the Information area of the negotiation banner to display logs and operating statistics for the Screen.
- Click the Logout button when you are finished configuring or monitoring the Screen.

Note – The Add New... choice list has text that is cut off when using a browser under Microsoft Windows for remote administration. This does not affect any operations and the system can continue to be used.

The Add New choice list should display the following items:

- Generate Screen Certificate...
- Load Issued Key Certificate...
- Load Issued Public Certificate...
- Associate MKID...
- New Group...

The following figure shows the Welcome page.

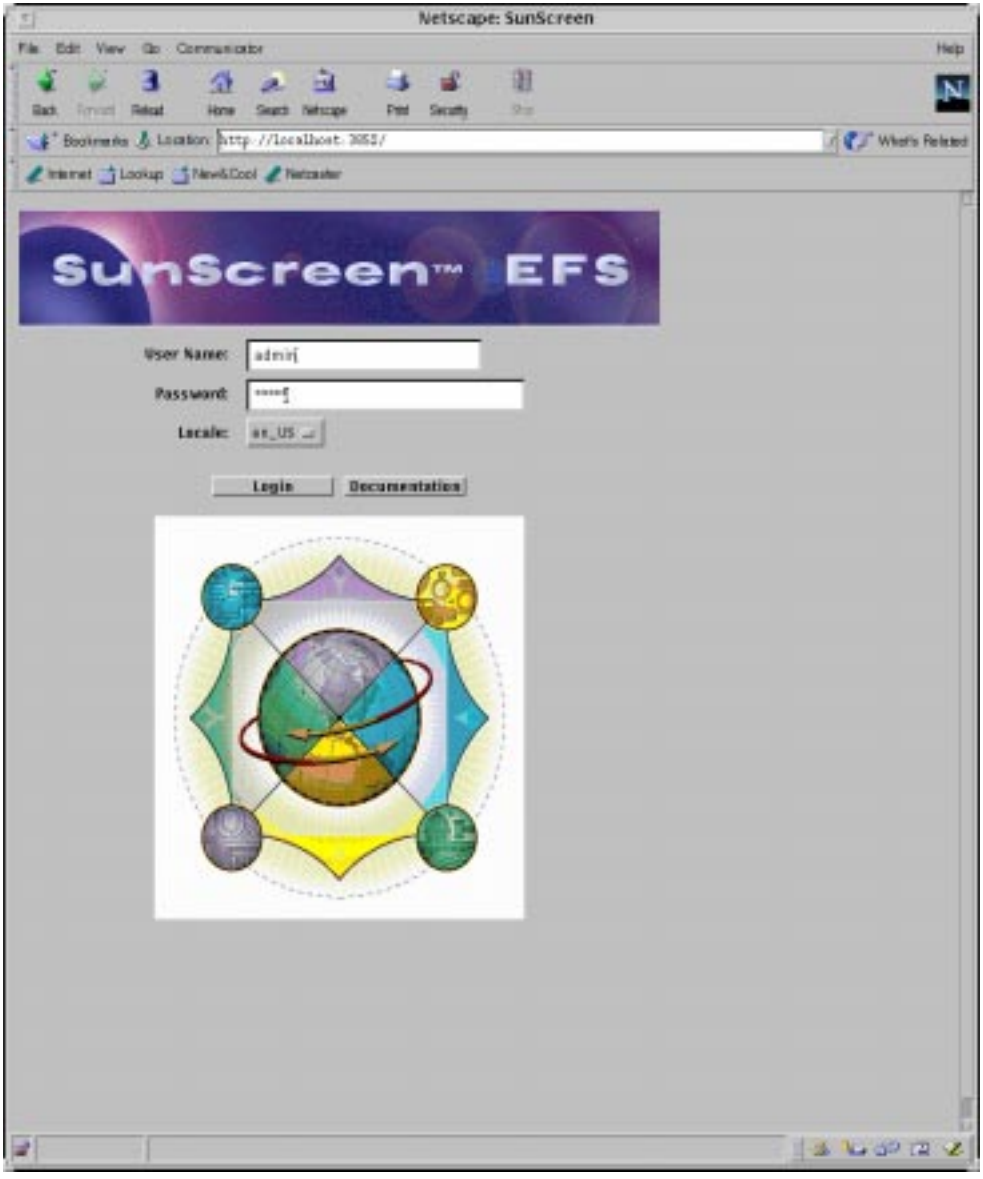


FIGURE 3-1 SunScreen EFS 3.0 Welcome Page

The following table describes the fields in the Welcome page.

TABLE 3-1 Welcome Page Fields

Fields	Description
User Name	Type your user name. The default user name is `admin`.
Password	Type the password associated with your user name. The default user password is `admin`.
Locale	Select the locale for the Screen. The default is `en_US`.
Login	Opens the SunScreen EFS 3.0 administration graphical users interface Policy List page.

Administration GUI Negotiating Buttons

The top-most area of the Policies List page contain buttons for Logout,Policies, Information, and Documentation. These buttons are explained the in the following table.

TABLE 3-2 Negotiating Buttons

Buttons	Description
Logout	Log out of the SunScreen EFS 3.0 session, which clears any lock you may be holding.
Policies	Open the Edit Rules page, where you create and edit the policy rules for SunScreen EFS 3.0.
Information	Open the Information page, which displays product information and HA status.
Documentation	Open the Documentation page, which contains links to the online SunScreen EFS 3.0 documentation.

Back and Forward Buttons

Use the Back and Forward buttons in the browser negotiation banner to move from the current page to either a previous or a next page, as available.

Documentation

Online documentation for SunScreen EFS 3.0 can be accessed by clicking the Documentation button on the administration GUI negotiating buttons.

Help System

The Help button displays context-sensitive help for the page you are on. It brings up a new browser window, which you can quit to get back to your page or move it aside and keep it open for quick reference.

Policies List Page

The Policy List page identifies the policies that have been stored for a Screen. The Policies List page allows you to add, copy, rename, verify, delete, backup, and restore your policy (see the Administration Guide for procedures). The following figure shows the Policies List page.

FIGURE 3-2 Policies List Page

Note – Changes made to the Policy List page take effect immediately. You cannot use the Revert Changes button to undo changes you make in this page.

The following table describes the controls available on the Policy List page.

TABLE 3-3 Controls on the Policy List Page

Control	Description
Policy list	Lists the policies that have been set up for your SunScreen EFS 3.0. You can manipulate Screen policies by selecting an entry in the Policy List and then clicking one of the buttons in the Policy List page.
Add New button	Opens a dialog window that prompts you for the name of the policy you want to add.
Edit button	Opens the Policy Edit page and allows you to manipulate the policy.
Copy button	Opens a dialog window that prompts you for the name of the policy to which you want to copy the information in the selected policy.
Rename button	Opens a dialog window asking for the new name you want to assign to the selected policy.
Delete button	Opens a dialog window asking you to confirm you want to delete the selected policy.
Activate button	Activates the selected policy for the Screen. After you click the Activate rule, the Config Name field in the Active Policy page turns green.
Initialize HA button	Opens the Initialize HA page.
Help button	Opens the topical help for Policy List page.
Backup All button	Opens the Backup All window, which lets you copy the SunScreen EFS 3.0 policy to a file or diskette. You cannot use the Backup All button you are using a browser whose security restrictions do not allow access to the file system from applets.
Restore All button	Opens the Restore All window, which lets you restore a SunScreen EFS 3.0 policy from a file or diskette. You cannot use the Restore All button if you are using a browser whose security restrictions do not allow access to the file system from applets.
Version -> button	Lists the different policy versions set up for your system. When you click on the Version -> button, it shows the different versions available for the selected policy, and changes to a left (<-). To remove the version information, click again and it reverts back to a right (->).

Policy Edit Page

The top area, below the Logout, Policies, Information, and Documentation negotiating buttons, is the Common Objects area where you control the policies global and Screen-specific objects. The bottom area is the Policy Rules area where you establish rules for Packet Filtering, Administrative Access, NAT, or VPN.

The following figure shows the Policy Edit page.

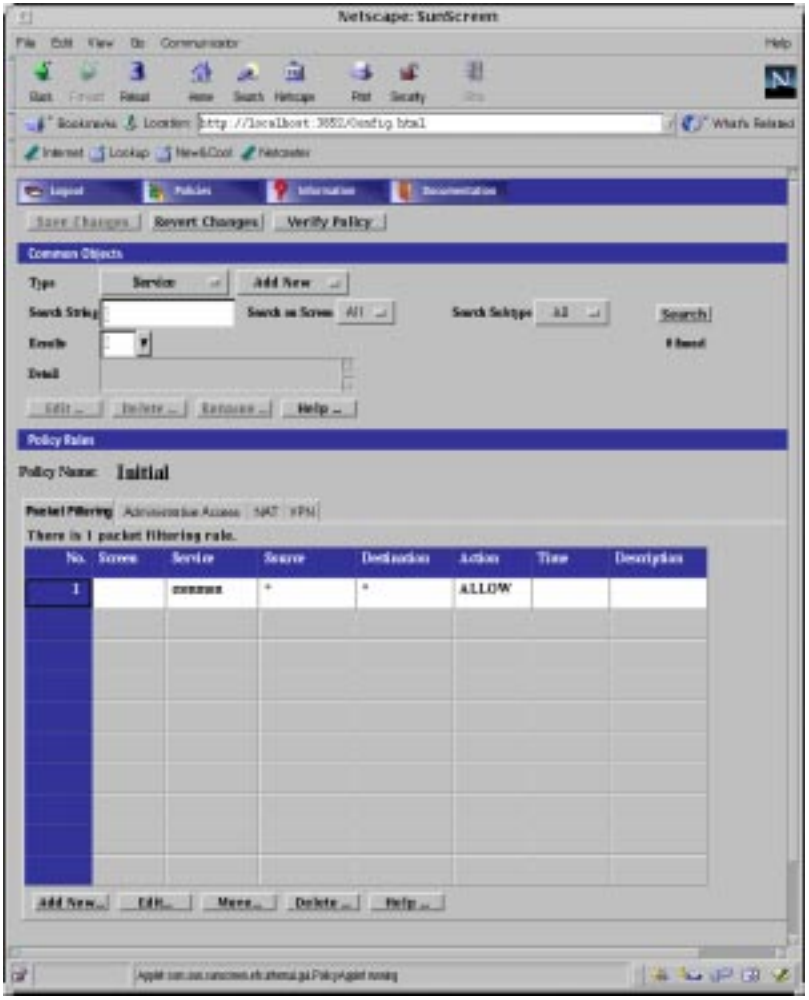


FIGURE 3-3 Policy Edit Page

Note – The choice list for the Action button disappears on Screens with low graphics resolution. A similar problem is seen when resizing the window, for example, the Edit Rule window. The problem goes away when you click the Cancel button and then reopen the window. You can type the first letter of the keyword in the ACTION combo box, such as the uppercase first letter (A, D, S, or E). The system automatically completes the keyword as the first letters are all unique. There are only four known keywords, which are all case sensitive: ALLOW, DENY, SECURE, and ENCRYPT.

Common Objects Area

Common objects are the components you use to make up policy rules. Before you write these rules, you add the common objects that you plan to use in the rules.

Note – After the common objects have been added, they are stored in a database and can be used over again to create rule sets for additional policies.

Save Is Not Required With Some Common Objects

Some of the common objects that appear in the administration GUI 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 applies immediately and is not revertible. The reason the Save button in the administration GUI does not become active when you alter these is because there is no in-memory buffer of unsaved changes.

Note – Although the changes made to these objects are saved immediately, they do not (except for `logmacro`) take effect until the next activation. The administration GUI edits Authorized Users, which are `authuser` objects, Proxy Users, which are `proxyuser` objects, and Jar Signatures and Jar Hashes, which are both `vars` objects.

Common Objects with this behavior are:

- Authorized User
- Admin User
- Jar Signature
- Jar Hash
- Proxy User

Common Objects

- Service — Policy object that lets you identify network services for which you want SunScreen EFS 3.0 to filter packets. You can define:
  - Single Service — Lets you add new network services and control the filtering activities applied when a service is used in a rule. 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.

    Use the fields in the dialog window to type the configuration information for the Service.

    Name — Specifies the name of the service object.

Note – If you are adding a new Service Object, either through command-line or the administration GUI and specify that you want to set the PARAMETERS but do not provide the information for these paramaters, the configuration editor dumps core. (You cannot specify a negative value for one of the parameters.) If you are using the administration GUI when this happens, it can hang as it cannot recover from the failed editing session. Make sure you always specify a PARAMETER value when specifing the use of parameters for a new service.

Description — (Optional) Provides a brief description about the service object.

Screen — Specifies which Screen recognizes the service object.

Use the Filter table to define the filtering activities:

Filter — Identifies the packed filtering engine.

Port — Identifies the port, program number, or type used by the forward filter.

Broadcast — Determines whether the rules where the service is used allows communication to broadcast and multicast addresses.

Note – If you want the service to work for non-broadcast addresses, type separate table entries for broadcast and non-broadcast filtering.

Parameters — Identifies the port, program number, or type used by the Reverse filter.

Reverse — Determines whether the filter applies to packets originating from the host in the To address of a rule and going to the From address of a rule.

- Service Group — Is a group of single services that you want to use together.

  Use the fields in the dialog window to type the configuration information for the service group.

  Name — Specifies the name of the service object.

  Description — (Optional) Provides a brief description about the service object.

  Screen — Specifies which Screen recognizes the service object.

Adding services to a service group:

1.Highlight the services you want in the Services list.

2.Choose Add to add the service to the Members list.

- Address — The address object for a range of addresses lets you associate a contiguous range of IP addresses with an address object name. For example, you can associate a name with a specified network address range and then use that name to filter traffic to all hosts on that network.

  Use the fields in the Address dialog window to type the configuration information address object.

  Name — Specifies the name for the address object.

  Description — (Optional) Provides a brief description about the address object.

  Screen — Specifies which Screen recognizes the address object. The default is All.

  Starting IP Address — Specifies the starting IP address in the range.

  Ending IP Address — Specifies the ending IP address in the range.

  - Single Host

    The address object for a single host lets you associate an individual host's address with the address object name in the Name field. Use the fields in the dialog window to type the configuration information for the rule.

    Name — Specifies the name for the address object.

    Description — (Optional) Adds a note about the address object.

    Screen — Specifies which Screen recognizes the address object. Default is All.

    IP Address/Host Name — Specifies the IP address of the host machine. To type an IP address in the address field you can do one of the following: Type the address directly into the field, or type the host name and use Lookup IP Address to get the host's IP address.

  - Address Range

    The address object for a range of addresses lets you associate a contiguous range of IP addresses with an address object name. For example, you can associate a name with a specified network address range and then use that name to filter traffic to all hosts on that network.

    Use the fields in the Address dialog window to type the configuration information address object.

    Name — Specifies the name for the address object.

    Description — (Optional) Provides a brief description about the address object.

    Screen — Specifies which Screen recognizes the address object. Default is All.

    Starting IP Address — Specifies the starting IP address in the range.

    Ending IP Address — Specifies the ending IP address in the range.

  - Address Group

    The address object for a group of addresses lets you group host addresses, address ranges and other address groups. By grouping addresses that use similar services and have similar actions, you can use groups to save time when creating rules.

Note – Before you create an address group, you first define the address objects-- single addresses, address ranges, or address groups-- that you want to use in the address group.

Use the fields in the dialog window to type the configuration information for the address object.

Name — Specifies the name for the address object.

Description — (Optional) Provides a brief description about the address object.

Screen — Specifies which Screen recognizes the address object. The default is All.

Addresses — Displays the addresses objects that can to be used to create the address group.

Include List — Specifies the address objects that are currently included in the address group. Use the Add or Remove buttons to modify the list.

Exclude List — Specifies the address objects that are excluded from the address group. For example, you can create an address group that includes all addresses except as specified in the Exclude List. Use the Add or Remove buttons to modify the list.

- Certificate — Certificate is a Common Object that allows you to manage certificates. You can:
  - Generate Certificate

    To generate a certificate, use the fields in the dialog window to type the configuration information.

    Name — Specifies a name for the certificate.

    Description — (Optional) Provides a brief description about the certificate object.

    Screen — Specifies which Screen recognizes the Certificate Object. The default is All.

    Installed On — (Optional) Specifies the Screen on which the certificate is installed.

    Radio buttons — Specify the level of encryption that the Screen uses.

    Generate New Certificate — Generates the certificate. The Certificate ID field displays the certificate's certificate ID.

  - Associate MKID

    Also called the certificate ID, lets you assign a name to a certificate that exists on another machine. You associate a certificate ID when you want to encrypt communication between two screens or between a screen and an Administration Station.

    Use the fields in the dialog window to type the information for associating the MKID.

    Name — Specifies the name for the certificate ID object.

    Description — (Optional) Provides a brief description about the certificate ID object.

Screen — (Optional) Specifies which Screen recognizes the certificate ID object. The default is all. Specifying a Screen allows you to define packet filtering rules that encrypt traffic between any two machines, not just between an Administration Station and a Screen.

Certificate ID — Specifies the certificate ID (hash value) for the certificate that you generated on the other system.

- Add new Associate MKID

  Associate MKID (also called the certificate ID) lets you assign a name to a certificate that exists on another machine. You associate a certificate ID when you want to encrypt communication between two screens or between a Screen and an Administration Station.

  Use the fields in the dialog window to type the information for associating the certificate ID.

  Name — Specifies the name for the certificate ID object.

  Description — (Optional) Provides a brief description about the certificate ID object.

  Screen — (Optional) Specifies which Screen recognizes the certificate ID object. The default is all.

Note – Specifying a Screen allows you to define packet filtering rules that encrypt traffic between any two machines, not just between an Administration Station and a Screen.

Certificate ID — Specifies the certificate ID (hash value) for the certificate that you generated on the other system.

- Screen — Allows you to edit or add Screen objects. You can edit miscellaneous Screen parameters, SNMP parameters, and Mail Proxy parameters for Screen objects that already exist.

  In general, you edit rather than create Screen objects because they are automatically created during installation. However, you create new Screen objects to configure HA and centralized management.

Note – You must enter the name of the Admin Interface of the Screen as listed in the Naming Service or in the hosts file.

Create a Screen object if you are setting up an HA cluster or centralized management group.

- HA / Master Config tab

  The HA / Master Config tab allows you to associate a certificate object with a Screen or Administration Station that is part of an HA cluster or a centralized management group. The HA menu and the Primary Name menu determine the role a Screen has within an HA cluster and centralized management group. The

settings you choose for these menus determines which other controls on the HA/Master Config tab are active. The meanings and uses of the fields in the HA/Master Config tab are as follows:

High Availability — Specifies whether the Screen is used for HA. If you are using it for HA, you can specify whether the Screen is a primary HA Screen or a secondary HA Screen.

Primary Name — Specifies the name of the primary Screen.

Administrative IP — Address IP address of the Screen that is used for administration.

Administration Certificate — Specifies the Administration Station's certificate name.

High Availability IP Address — Specifies the IP address of the HA interface.

Ethernet Address — Generated by the system.

Key Algorithm — Specifies the issued certificate (key) encryption algorithms supported for SunScreen SKIP, version 1.

Data Algorithm — Specifies the data encryption algorithms supported for SunScreen SKIP, version 2.

MAC Algorithm — Specifies the MAC (authentication) algorithms supported for manual keying.

Note – The entry in the Name field must be the same as the entry that exists in the `nameservice` lookup or in the `/etc/hosts` file. The IP address associated with this name must match the IP address of the administrative interface.

Note – The type of interfaces must be the same on all the machines in the HA cluster. This interface must be dedicated on each machine in the HA cluster with a dedicated network connection. For reasons of security, the HA network should not be connected to any other network. The HA primary Screen is always the Screen you administer whether it is the active or passive Screen.

- Editing a Screen object

  Use the Miscellaneous, SNMP, and Mail Proxy tabs in the Screen dialog window to edit a Screen object.

Note – After configuring the Screen as an HA primary remotely administered by an Administration Station (it is not part of the Centralized Management Group), the administration GUI does not show the admin certificate in the "HA/Master Config" panel from the Screen object. If you click OK from that panel, the admin certificate information is deleted from the Screen object. The result is that the activation fails because of the missing admin certificate. Use the command line to make any needed changes to the primary HA Screen object.

- Miscellaneous Screen parameters

  The Miscellaneous tab in the Screen dialog window allows you to specify miscellaneous Screen parameters.

  Log Size

  Stealth Network Address — Specifies the network address for interfaces that are used as SPF interfaces. Type this parameter if you have used the Interface object to designate any Screen interfaces as Stealth interfaces.

  Stealth Netmask — Specifies the netmask for interfaces that are used as SPF interfaces. Type this parameter if you have used the Interface object to designate any Screen interfaces as Stealth interfaces.

  Routing — Specifies whether the Screen is used for routing.

  Name Service — Specifies the name service that the Screen uses.

  Certificate Discovery — Specifies whether the Screen uses Certificate Discovery.

- SNMP

  The SNMP tab in the Screen dialog window allows you to add, edit, or delete an SNMP trap receiver.

Note – Use the Action field of the packet filtering Rule Definition dialog window to specify actions that generate SNMP alerts. The machine that you want to receive SNMP trap alerts must not be a remote Administration Station.

- Mail Proxy

  The Mail Proxy tab in the Screen dialog window allows you to add, edit, or delete domains known to distribute unsolicited electronic mail (spam). You can define spam domains if you use an SMTP proxy.

- Centralized management

  Centralized management allows you to remotely administer configurations on a group of Screens. A centralized management group comprises a primary Screen and a number of secondary Screens. The primary Screen's function is to "push" policy configurations to the secondary Screens in the group.

Note – You can configure packet filtering rules to make a specific rule apply to only one Screen. The rules apply globally by default.

Note – The entry in the Name field must be the same as the entry that exists in the `nameservice` lookup or in the `/etc/hosts` file. The IP address associated with this name must match the IP address of the administrative interface.

- Interface — A Common Object that lets you define interfaces and specify the actions a Screen should take when a packet, which is received on that interface, is rejected.

  Use the fields in the dialog window to type the configuration information for the interface object.

  Name — Specifies the name for the address object.

  Type — Specifies the type of interface.

  Screen — Specifies which Screen recognizes the Interface.

  Address Group — Specifies the valued source IP addresses for this interface.

  Logging — Identifies the level of detail for log messages generated when a packet received on the interface is rejected.

Note – You can override the level of log messages that are issued for individual rules.

  SNMP Alerts — Indicates whether SNMP alert messages are issued when a packet received on the interface is rejected.

  SNMP traps — Addressed to the defined hosts.

  ICMP Action — Identifies the ICMP rejection message that is issued if a packet received on the interface is rejected. The default is none.

  Comment — (Optional) Provides a descriptive note about the Interface object.

- Proxy User — A Common Object that contains the mapping information for users of SunScreen EFS 3.0 proxies. The FTP and telnet rules reference the proxy user entries.

  Before you add a proxy user, define an authorized user. You must create entries for both authorized users and proxy users to use the authentication feature of the FTP and telnet proxies.

  Name — Specifies the name of the proxy user.

  Description — Adds a brief description of the proxy user.

  User — Enabled Controls whether the user can log into the Screen.

  Authorized User Name — Selects the name of the authorized user to be used to authenticate this proxy user. Names in this list are generated when you add an Authorized User Object. If this field is empty, authorization is not required for this user.

- Authorized User — A Common Object that lets you specify which users are allowed to use the telnet and FTP proxy.

  Use the fields in the dialog window to type the configuration information for the Authorized User Object.

User Name — Specifies the login name of the authorized user.

Description — (Optional)Provides a brief description about the authorized user.

User Enabled — Indicates whether the user can log into the Screen.

Password — Specifies the login password for the authorized user.

Retype Password — Specifies the login password for the authorized user. The password typed in this field must exactly match the password you typed in the Password field.

SecurID Name — Specifies the user's login name for SecurID authorization.

Real Name — Identifies the real name of the authorized user.

Contact Information — Displays information on how to contact the specified user.

- Administrative User — A Common Object that lets you identify the administrators authorized to access the Screen.

  Use Administrative User Objects when you define Administrative Access rules.

  Use the fields in the dialog window to type the configuration information for the Administrative User.

  User Name — Specifies the login name of the administrative user.

  Description — (Optional) Provides a brief description about the administrative user.

  User Enabled — Indicates whether the user can log into the Screen.

  Password — Specifies the login password for the administrative user.

  Retype Password — Specifies the login password for the administrative user. The password typed in this field must exactly match the password you typed in the Password field.

  SecurID Name — Specifies the user's login name for SecurID authorization.

  Real Name — Identifies the real name of the administrative user.

  Contact Information — Displays information on how to contact the specified user.

  After you create the Administrative User Object, you grant administrative access by creating a rule in the Administrative Access panel.

Note – The name that you create for the Administrative User object is the same name that you use when you create an Administrative Access rule.

- Jar Signature — Lets you identify the Java archives (JARs) you want the Screen to pass. JAR signatures apply only to the HTTP proxy.

  Use the dialog window to type the information for the JAR signature.

  Certificate Name — Identifies the name of the certificate.

  Master Key ID — Identifies the certificate ID.

  Load Jar Certificate — Loads the certificate used to authenticate the Java archive.

- Jar Hash — The HTTP proxy can be set up to filter the Java applets based on the Jar hash value of the Jar file.

  Use the fields in the dialog window to type the information for the Jar hash.

Certificate Name — Identifies the name of the certificate.

Master Key ID — Identifies the certificate ID.

- Time — A Policy Object that lets you specify when a rule applies. You can specify the time of day and day of the week.

  Use the fields and controls in the dialog window to type the configuration information for the Time Object.

  Name — Specifies a name for the Time Object.

  Description — (Optional) Adds a descriptive note about the Time Object.

  Screen — Specifies which Screen recognizes the Time Object.

Note – The Time Object is based on a 24-hour clock. The time 00:00 is midnight on the day specified; the time 24:00 is midnight 24 hours later.

Policy Rules Page

SunScreen EFS 3.0 uses ordered sets of rules to implement the security policies for your site. *Ordered set of rules* means that, when a Screen receives a packet, it matches the packet against each rule in its active policy until it finds one that applies, and then takes the actions associated with that rule. Once a Screen finds an applicable rule for a packet, it ignores subsequent rules in its active policies.

Rules do not take effect until you save and activate the policy to which they belong by clicking the Save Changes button located above the Common Objects area, and the Activate button in the bottom area of the Policy List page. SunScreen EFS 3.0 sets up the basic security policy for name service lookups, Routing Information Protocol (RIP) packets, and SunScreen SKIP certificate discovery during installation. You can use the Rules area on the Policy Rules page to specify how your Screen should filter other types of packets.

Note – If the specified filtering rule fails to detect any issued certificate (key) encryption algorithms, it may display the following error message:
`An error occurred in detecting the Encryption algorithms. Please check if skipd process is running.`
If this occurs, use the `skipd_restart` command to restart the `skipd` process.

The Policy Rules page lets you add or modify a rule in the SunScreen EFS 3.0 policy. It opens when you click the Add New button (or when you select a rule and click the Edit button) in the Common Objects page.

Some pages use *tabs* to organize sets of related controls. To display the controls on a tab, click the tab header. The rules area contains four tabs: Packet Filtering, Administrative Access, NAT, and VPN as described in the following sections. The following table describes the four tabs.

The following table describes the tabs that are available from the Policy Rules page.

TABLE 3-4 Policy Rules Page Tab Items

Tab	Description
Packet Filtering	Shows the packet filtering rules(s).
Administration Access	Defines access rules for local administration and remote Administration Stations through the administration GUI or the command line (see Appendix B).
NAT	Maps private network addresses to public network addresses.
VPN	Maps name, address, certificate, issued certificate (key) algorithm, data algorithm, MAC algorithm, tunnel address, and description.

Packet Filtering Rules

The Packet Filtering tab brings up a panel that allows you to configure packet filtering rules. Use packet filtering to control traffic using a particular service, traffic intended for a particular service, or traffic coming from a particular address.

SunScreen EFS 3.0 uses ordered packet filtering. The Screen assumes that the first rule that matches a packet is the rule that governs the disposition of the packet.

If the packet does not match any rule, the Screen uses its default action to determine the disposition of the packet. Typically, the default action logs the packet and drops it, though other options are available.

The following table describes the available fields in the Packet Filtering tab.:

- *Rule Index (No)* — (Optional) Assigns a number to a rule. By default, this field displays a number one greater than the last rule (indicating this rule will be placed at the bottom of the list). If you type a lower number, the new rule is inserted into the specified position in the list, and the rules currently in the configuration are renumbered.
- *Screen* — (Optional) Specifies the Screen for which you want the rule to apply. Type a specific Screen name in this field if you use centralized management and want a rule to apply to a specific Screen.
- *Service* — Identifies the network service or service group to which this rule applies. Network services and service groups are described in Appendix B, “Services and State Engines.”

- *Source* — The value to which the source address of a packet is compared. If an asterisk (*) appears, any source address meets the criteria of the rule.
- *Destination* — The value to which the destination address of a packet is compared to determine whether the rule should apply. If an asterisk (*) appears, any destination address meets the criteria of the rule.
- *Action* — Displays the action for the rule: ALLOW, DENY, ENCRYPT, and SECURE.
- *Time* — Specifies the time of day for the rule.
- *Description* — (Optional) Provides a brief description of the Administrative Access rule.

Administrative Access Rules

Administrative Access rules allow you to specify access and encryption settings for local and remote SunScreen EFS 3.0 administrators.

Local Administrative Access

The Local Access Rules dialog window lets you add or modify administrative access rules for local Administration Stations. Use the fields in the dialog window to type the configuration information for the rule.

- *Rule Index (No)* — (Optional) Assigns a number to a rule. By default, this field displays a number one greater than the last rule (indicating this rule will be placed bottom of the list). If you type a lower number, the new rule is inserted into the specified position in the list, and the rules currently in the configuration are renumbered.
- *Screen* — (Optional) Specifies the Screen for which you want the rule to apply. Type a specific Screen name in this field if you use centralized management and want a rule to apply to a specific Screen.
- *User* — Lists the user names of SunScreen EFS administrators. Use the names that you defined for the Administrative User object.
- *Access Level* — Specifies what actions the designated user can perform:
  - ALL — Allows administrator to display and modify all setting for the Screen.
  - WRITE — Administrator can perform all operations except modifying the Administration Access rules for any Policy.
  - READ — Administrator can view both the Information and Policy. This level also allows the user to save and clear logs on the information page. With this access level users cannot modify any Policy data.
  - STATUS — Administrator can display status information (logs, statistics, status information) but cannot display or modify management settings.
  - NONE
- *Description* — (Optional) Provides a brief description of the Administrative Access rule.

Remote Administrative Access

The Remote Access Rules dialog window lets you add or modify administrative access rules for remote administration stations. Use the fields in the dialog window to type the configuration information for the rule.

- *Rule Index (No)* — (Optional) Assigns a number to a rule. By default, this field displays a number one greater than the last rule (indicating this rule will be placed bottom of the list). If you type a lower number, the new rule is inserted into the specified position in the list, and the rules currently in the configuration are renumbered.
- *Screen* — (Optional) Specifies the Screen for which you want the rule to apply. Type a specific Screen name in this field if you use centralized management and want a rule to apply to a specific Screen.
- *Address Object*
- *User* — Lists the user names of SunScreen EFS administrators. Use the names that you defined for the Administrative User object.
- *Encryption* — Specifies the version of SunScreen SKIP being used to encrypt traffic between the Screen and the Administration Station.
- *Certificate Group* — Specifies the name of the certificate group allowed in over this interface, which can correspond to a single certificate or a certificate group.
- *Key Algorithm* — Identifies the algorithm used to encrypt traffic-encrypting issued certificates (keys). The algorithms available depend on the version of SunScreen EFS (U.S.&Canada, Export Controlled, or Global) you are using.
- *Data Algorithm* — Identifies the algorithm used to encrypt message traffic between the Screen and the Administration Station. The algorithms available depend on the version of SunScreen EFS (U.S/Canada,Export Controlled, or Global) you are using.
- *MAC Algorithm* — Identifies the algorithm used to authenticate traffic.
- *Tunnel* — Identifies the Tunnel address used for the communication between the remote Administration Station and the Screen.
- *Access Level* — Specifies what actions the designated user can perform:
  - ALL — Administrator can display and modify all setting for the Screen.
  - WRITE — Administrator can perform all operations except modifying the Administration Access rules for any Policy.
  - READ — Administrator can view both the Information and Policy. This level also allows the user to save and clear logs on the information page. With this access level users cannot modify any Policy data.
  - STATUS — Administrator can display status information (logs, statistics, status) but cannot display or modify management settings.
  - NONE
- *Description* — (Optional) Provides a brief description of the remote administrative access rule.

NAT Rules

The Network Address Translation (NAT) tab allows you to set up mapping rules to translate IP addresses according to specific rules. These rules interpret the source and destination of incoming IP packets, then translate either the apparent source or the intended destination, and send the packets on. You can map hosts, lists of addresses, ranges of addresses, or specific groups, depending on what you have configured in your EFS 3.0 installation. See Address Common Object to define addresses, ranges, or groups of addresses.

In general, you would map addresses to:

- Ensure that internal addresses appear as registered addresses on the Internet, or
- Send traffic for a specific destination to a different, pre-determined destination.

It is not possible to translate both source and destination addresses-- that is, to make packets appear to come from a different IP address and to simultaneously direct the packets to a different destination.

When defining NAT rules, the first rule (lowest number) that matches a packet wins, and no other rules can apply. Therefore, you might define specific rules first, then broader cases later.

The meanings and uses of the specific fields in the NAT screens are as follows:

- *Rule Index (No)* — Assigns a number to a rule. By default, this field displays a number one greater than the last rule (indicating this rule will be placed at the end of the list). If you type a specific number, the new rule is inserted into that position in the list, and the rules currently in the configuration are renumbered.
- *Screen* — Specifies the Screen for which you want the rule to apply. Type a specific Screen name in this field if you use centralized management and want a rule to apply to a specific Screen.
- *Mapping*
  - Static — Specify static mapping to set up a one-to-one relationship between two addresses. You could use this to set new apparent IP addresses for hosts on your network without having to reconfigure each host, for example.
  - Dynamic — Specify dynamic mapping to map source addresses to other addresses in a one-to-many relationship. You could use dynamic mapping to ensure that all traffic leaving the firewall appears to come from a specific address or group of addresses, or to send traffic intended for several different hosts to the same actual IP access.
- *Source* — Specify the source address to map from an untranslated packet. Source addresses are the actual addresses contained in the packet entering the firewall.
- *Destination* — Specify the untranslated destination address for the source packet. Destination addresses are the actual addresses contained in the packet entering the firewall.
- *Translated Source* — Specify the translated source address for a packet. The translated source is the address the packet appears to originate from.

- *Translated Destination* — Specify the translated destination packet address. The translated destination is the actual address the packet goes to after it leaves the firewall.
- *Description* — Used to provide a description of the mapping defined in this rule.

As you define rules, remember that you cannot translate both source and destination addresses. You must either translate packets so they appear to come from a different source, or translate packets so they go to a specific destination, but not both.

All NAT rules are unidirectional -- that is, they work precisely as defined, and are not interpreted as also applying in the reverse direction. If you want rules to apply in both directions, you must specify two different rules. For example, if you map a source address from `internalname.com` to the destination of `publicip.com`, you will also have to map a source of `publicip.com` to the destination of `internalname.com` to translate traffic in both directions.

Virtual Private Network (VPN) Gateway Rules

The VPN tab allows you to define Virtual Private Network (VPN) gateways. Defining VPN gateways using this mechanism simplifies the creation of VPNs that include more than two gateways.

Note – Each gateway in this type of configuration must be able to connect to the other ones directly--without going through another gateway.

Defining VPN Gateways

Use the fields in the VPN dialog window to define VPN gateways:

- *Rule Index (No)* — (Optional) Assigns a number to a rule. By default, this field displays a number one greater than the last rule (indicating this rule will be placed bottom of the list. If you type a lower number, the new rule is inserted into the specified position in the list, and the rules currently in the configuration are renumbered.
- *Name* — Specifies the Name of the VPN to which this gateway belongs.

Note – Type the same name in the Name field for each gateway that you include in the VPN.

- *Address* — Specifies the machine to be included in the VPN.
- *Certificate* — Specifies the name of the certificate for this VPN gateway.

- *Key Algorithm* — Specifies the issued certificate (key) algorithm the VPN uses. All gateways in the same VPN must use the same issued certificate (key) algorithm.
- *Data Algorithm* — Specifies the data algorithm the VPN uses. All gateways in the same VPN must use the same data algorithm.
- *MAC Algorithm* — Specifies the MAC algorithm the VPN uses. All gateways in the same VPN must use the same MAC algorithm.
- *Tunnel Address* — Specifies the destination address on the outer (unencrypted) IP packet to which tunnel packets are sent.
- *Description* — (Optional) Provides a short description of the VPN gateway.

Adding a VPN Rule

After you define the gateways in your VPN, add a Packet Filtering rule for this VPN. 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.

When you add a packet filtering rule for VPN, leave the Screen field empty.

- Specify SECURE for the packet filtering action.
- Type the name of the VPN in the VPN field.

CHAPTER 4

SunScreen EFS 3.0 Logs

SunScreen EFS 3.0 provides flexible logging of packets. You can configure SunScreen EFS 3.0 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 definition of a rule controls whether a packet is logged and what information about the packet is recorded.

Examining logged packets is useful when you are identifying the causes of problems during configuration or administration of your SunScreen EFS 3.0. You should also examine logs periodically for evidence of attempts to break into your network.

Logging Limitations

1. 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.

2. Decrypted packets are logged, but SKIP certificate IDs are not logged.

   SunScreen EFS 3.0 cannot guarantee the identity of who is coming through the firewall from an audit point of view.

3. Only the active system logs packets.

   When the active HA cluster Screen fails, its logs are lost, and the new active HA cluster Screen begins logging the packets.

Log File Locations

SunScreen EFS 3.0 stores its log files in the following locations:

- The SunScreen EFS 3.0 packet, session, and extended event log is stored on the Screen in the `/var/opt/SUNWicg/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

In SunScreen EFS 3.0, the size of the area used to log packet traffic, session and other events is configurable. The og file contains a number of files in a particular directory. Each Screen has a log file of its own, and the size of the log file on each Screen can be configured specifically.

Log file sizes are established in much the same way as other configuration items, and these sizes are propagated to various Screens being managed during the normal activation process. However, the actual resizing of the log file on a particular Screen only occurs on the next restart of that Screen after the activation that changes the size. This is true for primary and secondary Screens in a Centralized Management group and in an HA cluster.

It is advisable that changes to the log file size(s) be made during initial Screen installation because resizing of the log file on a particular Screen only occurs on the next restart of that Screen, after the activation that changes the size.

Note – Setting the size of the log file does not cause immediate allocation of the filesystem space to store the log. Hence, other competing users of the filesystem on which the log file resides should 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 is controlled by the variable *LogSize*. It contains the following items:

```
prg=log
name=LogSize
value=size                    (in Mbyte units)
description="descriptive text" (optional)
enabled | disabled            (default is enabled)
```

The global default log size can only be configured using the command line interface (see Appendix B).

Group-Screen installations are configured on the primary Screen.

The following is an example of what you type to display the global default log file size, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=log name=LogSize PRG="log" NAME="LogSize" ENABLED
VALUE="100" DESCRIPTION="global log capacity (MB)" ...
```

The following is an example of what you type to set the global default log file size, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars add prg=log name=LogSize value=new size description=new
description
edit> quit
```

Tip – It is not necessary to type `save` before `quit` above if only `authuser`, `proxyuser`, `logmacro`, or `vars` entities are altered.

The following is an example of what you type to set the global default log file size to 250 Mbytes, while logged in to the primary Screen:

```
edit> vars add prg=log name=LogSize 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 to protect 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).

The following is an example of what you see if you attempt to `save` without changing entities other than these types, you are reminded by a message:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message and you can simply `quit` the configuration editor.

Once a log file size has been changed, the system configuration must be activated to propagate the change. Then the effected Screen(s) must be restarted for the log file size change to take effect. (In the case of a change to the global default log file size, the effected Screens are all Screens except those for which a log file size has been specifically configured.)

Configuring the Log Size for a Specific Screen

Configuring the global log size for a centralized management group of Screens, or Screens in an HA cluster, is performed on the primary Screen through the administration GUI.

Global log size is also configured through the command line (see Appendix B). It is controlled by the variable *LogSize*.

The following is an example of what you type to display the log file size for a specific Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> list Screen scrn1 ADMIN_CERTIFICATE "scrn1.admin" CDP
ROUTING DNS scrn2 ADMIN_CERTIFICATE "scrn2.admin" CDP ROUTING
DNS LOGSIZE 444
```

`scrn1` does not have the log file size configured and so uses the global default value. `scrn2` has a size of 444 (Mbytes) that is used instead of the global default value on that Screen.

The following is an example of what you type to set the log file size for a specific Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> add Screen scrn1 ADMIN_CERTIFICATE scrn1.admin CDP ROUTING
DNS LOGSIZE 20
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.

The following is an example of how you would alter the log file size for a specific Screen through the administration GUI:

1. **From the Policies page, select and edit the policy to be altered.**
2. **Select the desired Type: "Screen common object."**
3. **Edit the Screen's common object.**
4. **Alter the Log Size entry through the Miscellaneous tab.**
5. **Save the change in Save Changes.**
6. **Activate the policy.**
7. **Restart the Screen for the log file size change to take effect.**

Configuring Events to be Logged

Logs contain three basic types of events:

- network traffic (packet)
- network session summaries
- extended events

For a given program component, the level of logging can be specified. This is done by means of a variable setting for that component; the name of the variable is *LogSeverity*. A variable that is specific to a particular Screen overrides the general setting for that component. Beyond the variable setting for a specific component, a

general (non-component-specific) variable controls otherwise unlimited logging; again, a variable that is specific to a given Screen overrides this general default. This search order can be summarized as:

Key Sought
`sys=`*Screenname* `prg=`*programname*
`name=`*LogSeverity*
`prg=`*programname* `name=`*LogSeverity*
`sys=`*Screenname* `name=`*LogSeverity*
`name=`*LogSeverity*

As initially configured, SunScreen EFS 3.0 contains variables defined for each program components logging variable, along with the non-component non-Screen (`global global`) default; all are initially set to the value "`info`".

Configuring events to be logged can only be configured using the command line interface.

Configuring Log Event Limiters

The log limiters are controlled by `LogSeverity` variables as previously introduced. Each such variable contains the following items:

`sys=`*Screenname*	(optional)
`prg=`*programname*	(optional)
`name=`*LogSeverity*	
`value=`*severityname*	(emerg,alert,...,debug)
`description="`*descriptive text*`"`	(optional)
`enabled	disabled`	(default is enabled)

The *LogSeverity* variables can only be configured using the command line interface (see Appendix B). For group-Screen installations, they are configured on the primary Screen.

The following is an example of what you type to display the `global global` log limiter, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars print name=LogSeverity
NAME="LogSeverity" ENABLED VALUE="INFO"
DESCRIPTION="global log severity limit" ...
```

The following is an example of what you type to display the global log limiter for authentication events, while logged in to the primary Screen:

```
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" ...
```

The following is an example of what you type to cause more (debugging) information to be logged on a particular Screen for authentication events, while logged in to the primary Screen:

```
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 to protect 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.)

The following is an example of the message you see if you attempt to `save` without changing entities other than these types:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message and you can simply `quit` the configuration editor.

Once log limiters have been altered, the configuration must be activated to propagate the changes.

Log Retrieval and Clearing

A Screens log is retrieved and can be cleared using the `log` sub-command of `ssadm`. Filtering can be applied by the Screen during retrieval, reducing the content of the log retrieved (see the section, “Log Inspection and Browsing”).

You can retrieve and post-process logs on an automated basis (details regarding automated log management can be found in the `man` page for `sslogmgmt(1M)`). The maintenance command `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 available through SUNWicgSA.

`log` has three sub-commands:

`get`	Retrieves the current content of the log, but leaves it intact.
`get_and_clear`	Combines these two functions, clearing the log through the point to which it was retrieved.
`clear`	Makes the current content of the log inaccessible for further retrieval operations.

Retrieval of the log produces the (binary) log records on the standard output of `ssadm`. This is either redirected to a file or perhaps piped into other processes, such as `ssadm logdump` for viewing, further filtering, and so forth. Manipulation of the output stream should be performed by the shell you are using on your administration platform.

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.

The following is an example of what you type to get the log from a given (primary or secondary) Screen to store in a local file:

```
admin% ssadm -r Screen log get [filterarg ...] > locallogfile
```

The following is an example of what you type to get and clear the log automatically:

```
admin% ssadm -r Screen log get_and_clear [filterarg ...] > locallogfile
```

The following is an example of what you type to simply clear the log:

```
admin% ssadm -r Screen log clear
```

The *get_and_clear* and *clear* verbs take an optional argument: a designation of who cleared the log. This is a text string that is displayed when log statistics are retrieved.

The following is an example of what you type to get and clear the log, setting this designator:

```
admin% ssadm -r Screen log -U "ben" get_and_clear [filterarg ...] > ...
```

The following is an example of what you type to simply clear the log, setting this designator:

```
admin% ssadm -r Screen log -U "Ben: reset after move" clear
```

Log Statistics

The current statistics of a Screens log file can be inspected in the administration GUI-based log browser.

These same statistics are also visible using the `logstats` sub-command of `ssadm`.

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` Sub-Command

The following is an example of what you type 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
```

The `Log size` item 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 (though minimal) space needed to manage those records.

The `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.

The `Cleared at:` gives the date and time the log was last cleared. If a text designator (through the `-U` option to the `log` verb) 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 Screens active log file can be browsed in either a historical or live mode.

Note – 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

Filtering Screen's logs employs a common filtering mechanism and language, regardless of the context in which it is used, that is embodied in the logdump command. logdump is based on, and is a superset of, the snoop program, which is provided with the standard Solaris operating system platform.

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, pre-defined filters can be employed to simplify and regularize routine log processing tasks.

The general usage for logdump is as a sub-command 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.

Reminder: although logdump is used directly as an ssadm sub-command, all other places in SunScreen where log filtering is allowed employ the same filter specification language. Hence, examples in this manual section should be viewed as 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.

The following is an example of what you type to process (piped-in) records from the standard input:

```
% ssadm -r Screen log get | ssadm logdump -i- [output args ...] [filter
args ...]
```

The following is an example of what you type to process local file log record input:

```
% ssadm logdump -i locallogfile [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, for example:

```
% ... ssadm logdump -iinput arg -o- [filter args ...] | ...
or:
% ... ssadm logdump -iinput arg -olocallogfile [filter args ...]
```

To output readable text, omit the `-o` option.

The formatting options for readable text are common to `snoop`; these are `-v`, `-V`, `-t[r|a|d]`, and `-x`*offset*`[,`*length*`]`.

`logdump` Extensions

`logdump` is an extension of the standard `snoop` packet monitoring tool provided with the Solaris operating system. In general, any expertise in the use of `snoop` is directly applicable to use of `logdump`.

The facilities of `logdump` that are common to `snoop` are not detailed here; refer to the `ssadm-logdump(1m)` man page.

`logdump` has been extended to provide for the special additional needs of the SunScreen system. These extensions are summarized as:

- Extensions to the format of network packets logged: inclusion of the interface on which logged packets arrive inclusion of the reason (why) packets are logged
- 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 objects
- Display of SKIP packet encapsulations
- Display of all extensions listed above

`logdump` is also fundamentally different from `snoop` in the respect that it is not involved in decisions as to what is logged by SunScreen (rules and variables previously described provide this control). Rather `logdump` serves as a means to post-process log file content only. (`snoop` is often used to filter network input during the process of capture or direct display.)

SunScreen logs and `snoop` capture files are not interoperable.

Logged Network Packet Enhancements

On SunScreen, the log contains network traffic that arrives on multiple link-layer interfaces in a con temporally-interspersed manner. For this reason, it is important to record the interface upon which the traffic was received. The interface is memorialized 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 the `snoop`-produced capture file.

Additionally, the SunScreen packet Screen is configurable to log network traffic for a variety of reasons, such as packets that were passed successfully, those that failed to match rules, those that arrived after session state expired, and so forth. This reason is recorded as an unsigned integer, commonly referred to as the `why` code. (See Appendix D, “Error Messages,” for a complete table of these reasons in “Logged Packet Reasons.”)

`logdump` displays these extended items and allows filtering based on these extended items.

The following is an example of what you type to restrict, based on interface, the `logiface`. 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, you would type:

```
% ... logiface qe ...
```

The following is an example of what you type to restrict based on the reason a packet was logged, the `logwhy` operator is provided. It 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, you would type:

```
% ... logwhy 1 ...
```

General Event Type Enhancements

In addition to network packet traffic, SunScreen 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 following is an example of what you type to restrict to network packet traffic events:

```
% ... loglvl pkt ...
```

Note – The `logiface` and `logwhy` operators imply `loglvl pkt`.

The following is an example of what you type to restrict to session summary events:

```
% ... loglvl sess ...
```

Note – In previous SunScreen releases, the `sas_logdump` program provided `-S` and `-s` options that provided a crude form of the `loglvl sess` feature. Those options are no longer supported.

The following is an example of what you type to restrict to authentication events:

```
% ... loglvl auth ...
```

The following is an example of what you type to restrict to application events:

```
% ... loglvl app ...
```

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:

```
% ... from src host ...
```

matches packet, session, and extended events that originated from the given source host.

Similarly, the filtering mechanisms inherited from `snoop` 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:

```
% ... port svc ...
```

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 allow a wide variety of events to be recorded in the log, both in SunScreen EFS 3.0 and into future releases. Because of the self-described syntax used, virtually any event can conceivably 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:

```
% ... 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:

```
% ... logapp ftpp ...
```

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, but are common in the sense that they are handled in a consistent way. These attributes are:

TABLE 4-1 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 (usu. 6 for TCP or 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 events mission in life.
`msg`	Generic message text.

Log Filtering Macros

For SunScreen EFS 3.0, 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 have the benefits of automatic propagation to secondary Screens that are enjoyed by 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 as a means to deal with emergency situations and other issues of expediency where the process of central macro definition and mass activation is unacceptable in the face of the conditions at hand. As a general practice, it is a good idea to 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 EFS 3.0. Evaluation mechanisms prefer a Screen-specific macro with a given name over a global one. Evaluation of macros occurs at the time of usage. (Users familiar with computer programming languages 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 hostname-to-address and service name-to-port number mapping mechanisms of `snoop`, namely, DNS, NIS, host and service tables defined for standard Solaris.

Log macros provide a way to access the name mappings of the registry in translating names to values.

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=Screen                    (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` sub-command of `ssadm`. For group-Screen installations, they are configured on the primary Screen.

The following is an example of what you type to display the definition of a non-Screen specific macro, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp"
DESCRIPTION="SMTP mail" ...
```

The following is an example of what you type to define a non-Screen specific macro, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro add name=pkts-only value="loglvl pkt"
Description="only network packets"
edit> quit
```

The following is an example of what you type to define a macro for a specific Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro add sys=Screenname name=SFO-routing value="port
rip src SFO-routers" description="routing activity in SFO
district"
edit> quit
```

Note – Although, the output produced by `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. 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.)

The following is an example of the message you see if you attempt to `save` without changing entities other than these types:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message and you can simply `quit` the configuration editor.

Log macros are available for immediate use on the Screen whereupon they have been defined. It is not necessary to do an activate each time a log macro is changed to use it. However, to propagate log macro definitions from a primary Screen to secondaries, activation is necessary.

As previously indicated, it is also possible to create expediency log macros on any Screen. This is done using `logmacro` as a sub-command of `ssadm` (rather than a `ssadm edit` sub-command). The syntax of the rest of the usage is the same as given above.

The following is an example of what you type to display the definition of a non-Screen-specific macro, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro print name=mail-only
NAME="mail-only" ENABLED VALUE="svc smtp"
DESCRIPTION="SMTP mail" ...
```

The following is an example of what you type to define a macro for a specific Screen, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro add sys=slave name=SFO-routing
value="port rip src SFO-routers" description="routing activity in
SFO district"
```

Caution – It is bad practice to define Screen-non-specific log macros on secondary Screens. In future SunScreen firewall releases, the ability to do so will be removed.

Log Macro Name and Body

The name of a log macro, as has been previously shown, 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`. It its simplest form, this is simply 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 found therein, 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` and, if their operand resolves in the service registry, the operator-operand sequence is rewritten with the registry value.

Note – In SunScreen EFS 3.0, 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 macros body.

From the discussion of the processing done on the `macrobody` string during evaluation, it should be evident why macro names must be simple identifiers. Similarly, expansion cannot handle addresses nor 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` sub-command of `ssadm edit`. Individual macro definitions can be displayed. Also, all Screen-non-specific definitions, or all definitions specific to a Screen, or all definitions specific to any Screen, can be displayed. An abbreviated listing containing just the names of these last three classes of macros can also be generated.

The following is an example of what you type to display a specific macro definition, while logged in to the primary Screen:

```
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-routing
SYS="secondary" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district" ...
```

The following is an example of what you type to display all Screen-non-specific definitions, while logged in to the primary Screen:

```
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" ...
```

The following is an example of what you type to display all definitions specific to a Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro print sys=slave
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to display all definitions specific to any Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro print sys=
SYS="master" NAME="HQ-routing" VALUE="port rip src HQ-routers"
DESCRIPTION="routing activity in HQ district" ...
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to produce name lists, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names sys=
SYS="master" NAME="HQ-routing" VALUE="port rip src HQ-routers"
DESCRIPTION="routing activity in HQ district" ...
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to display all Screen-non-specific names, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names
NAME="mail-only"
NAME="pkts-only"
```

The following is an example of what you type to display all names specific to a Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names sys=slave
SYS="slave" NAME="SFO-routing"
```

The following is an example of what you type to display all names specific to any Screen, while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> logmacro names sys=
SYS="master" NAME="HQ-routing"
SYS="slave" NAME="SFO-routing"
```

The following is an example of what you type to display a specific macro definition, while logged in to the primary Screen:

```
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=slave
name=SFO-routing
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to display all Screen-non-specific definitions, while logged in to the primary Screen:

```
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" ...
```

The following is an example of what you type to display all definitions specific to a Screen, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro print sys=secondary
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to produce name lists, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro names sys=secondary
SYS="slave" NAME="SFO-routing" VALUE="port rip src
SFO-routers" DESCRIPTION="routing activity in SFO district"
...
```

The following is an example of what you type to display all Screen-non-specific names, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro names
NAME="mail-only"
NAME="pkts-only"
```

The following is an example of what you type to display all names specific to a Screen, while logged in to the primary Screen:

```
admin% ssadm -r secondary logmacro names sys=secondary
SYS="slave" NAME="SFO-routing"
```

Log Macro Usage

A log macro is utilized by expanding its value and by causing that expansion to be presented as a filter expression to a `log get*` or `logdump` command.

The following is an example of what you type to perform log macro expansion using `logmacro` as a sub-command to `ssadm`, consider the following, while logged in to Screen:

```
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"
```

The above shows two macros defined. The first, `probed-ports` is Screen-non-specific and ostensibly defines services that are thought to be targets for initial probes leading to security attacks. The second, `suspicious`, is specific to `Screen` and contains a more complete macro for filtering potential probes; it restricts itself to packets logged because there was no rule found or they had source addresses that were illegal on their interface (“`logwhy 256`”), further to packets arriving on a specific (presumably outside) interface (“`logiface le0`”), yet further to packets originating from non-trusted hosts or targeted at hosts that are non-published (“not from trusted or to hidden”), and yet further to restrictions imposed by the macro “`probed-ports`”.

Tip – As a brief aside, the verb `names,flat` produces a list of names that are available for macro expansion on a particular Screen.

For example, while logged in to Screen, type:

```
admin% ssadm -r Screen logmacro names,flat
"probed-ports"
"suspicious"
```

Note – Screen-specific issues of macros have been hidden, listing macro names as they are used by embedded macro references.

Assuming 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
```

The following is an example of what you type to expand the given macro, while logged in to 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 whereas 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 facilitate simple command-line usage in conjunction with the other log processing facilities of SunScreen.

The following is an example of what you type to employ the above macro to retrieve the suspicious items in the current log on the Screen and display them, while logged in to Screen:

```
admin% ssadm -r Screen log get ‘ssadm -r Screen logmacro expand
suspicious‘ | ssadm logdump -V
```

CHAPTER 5

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.

SunScreen EFS 3.0 lets you 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 installation of any 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 script checks to see if the policy being activated contains one or more rules that utilize 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.

The following diagram shows a Screen using a proxy to filter packets for the HTTP protocol.

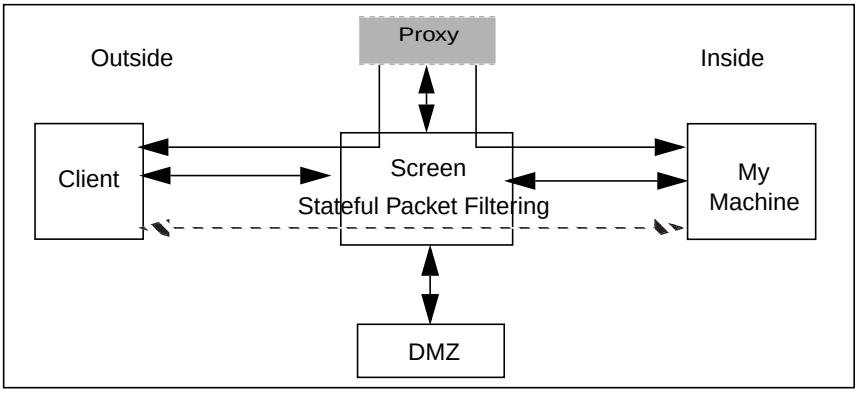


FIGURE 5-1 Screen With a Proxy

How Proxies Work

Each proxy is a multi-threaded program provided with SunScreen EFS 3.0. 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 EFS 3.0 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 sub-rules for each proxy-use rule; one that allows client access to the proxy server program, and the other that allows the proxy server program to access the backend server(s). You do not see these rules as they are hidden but 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, and content filtering as may be configured.

The general flow of the rule that tests each proxy applies to access requests for FTP and telnet proxies is, 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 the section entitled “User Authentication.”

Then, for each rule configured for a particular proxy:

1. Is the requested service port one handled by the proxy?

   This check is rather trivial, as the proxy only receives connection requests for the port(s) on which it is configured to listen.)

2. Is the address of the client contained in the set of source addresses for the policy rule?

3. Is the address of the backend server (if applicable) in the set of destination addresses for the policy rule?

4. For HTTP URL references with specific port numbers — is the target service port allowed by the proxy?

5. For FTP or telnet 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 and telnet proxies of SunScreen EFS 3.0 provide the ability to restrict access to users who can verify their authenticity.

User authentication mechanisms of SunScreen EFS 3.0 are described in detail in a subsequent major section “User Authentication.” 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 the establishment of 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 users userid on Solaris).

The second goal is achieved during 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 in SunScreen EFS 3.0 has the following limitations:

- You cannot set up an FTP proxy through the HTTP proxy. Consequently, internal users cannot use a browser for FTP file access through a Screen.
- Proxies can only be used on Screens with at least one routing-mode interface.
- You cannot use proxies on Screens in an HA cluster.

FTP Proxy

The FTP proxy functions as a relay for the File Transfer Protocol to let 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 (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 / 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(es) 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 (given above in the section “Policy Rule Matching”) is employed. 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 login using the backend username generated during authentication and using `serverpass` as the password (see “Proxy User Authentication” above). 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 2.0
    :Username to be given as proxy-user@FTP-server
    :Password to be given as proxy-pass@FTP-server-pass
220 Ready
Name (screen:edison): anonymous@ftp.cdrom.com
331-Proxy: Authenticate and connect
331 Password needed to authenticate anonymous.
Password:
    :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 Guest login OK, send your email address as password
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
```

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 users email address.

FTP Proxy Use

To use the proxy and make FTP connections, the user must open an FTP connection to the Screen (as shown in the preceding example) rather than open a direct connection to the end system. The Screens 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 employs a 10 minute timeout on the control connection for user requests. Its idle allows backend server responses 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` (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 type to display this (initial) variable while logged in to 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 may wish to alter this number of sessions, perhaps to be more restrictive on a particular Screen.

The following is an example of what you type to do this while logged in to the primary Screen:

```
edit> vars add sys=Screen prg=httpp 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 so as to exclude all of your Screen(s) addresses. (You can still FTP out of the Screen, as necessary.)

HTTP Proxy

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). The HTTP proxy allows or denies sessions based on the source and destination addresses. Additionally, the HTTP proxy provides selective filtering of HTTP content, such as Java applets, ActiveX, and cookies, based on the source and destination addresses for sessions. Finally, the HTTP proxy can filter Java applets based on the signatures encapsulated in Java Archive (JAR) files attached to the applet or based on a precomputed hash of valid applets.

To use the HTTP proxy, define the source addresses that should be allowed to access the proxy and 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).

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 to access services that would otherwise be restricted save this feature of HTTP.

The port restriction mechanism for the SunScreen EFS 3.0 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=httpp`
- `name=`*TargetSvcs*
- `values={ svc=`*service* `... }`(name(s) of service objects for services to allow)
- `description="`*descriptive text*`"`(optional)
- `enabled | disabled` (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 type to display this (initial) variable while logged into the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> vars print prg=httpp name=TargetSvcs
PRG="httpp" NAME="TargetSvcs" ENABLED VALUES={ svc="www" }
DESCRIPTION="target TCP ports that the HTTP proxy can get to"
```

You may wish to be able to access a wider range of ports in their URLs. One such configuration is obtained by configuring a new service object and an added variable (for example, this time, for a particular Screen) Again, the following is an example of what you type while logged in to 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=httpp 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 3851-3855 (the Screens administrative and SecurID PIN server ports), and ports 6000-6063 (used by X windows). Tailor your restrictions to suit your security needs.

The SunScreen EFS 3.0 HTTP proxy can also be configured to restrict Web content to be allowed through. Content can be blocked or allowed, or it can be configured to verify certain content (Java applets) based on digital signatures or hashes. These content-filtering features are configured as part of the rules employing the HTTP proxy.

HTTP Proxy Operation

When the HTTP proxy starts, it reads its policy files, starts a Java Virtual Machine (JVM) for processing JAR 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. It then searches the proxy policy rules for a match on the source and destination addresses.

- If a match is found, the flags associated with that policy rule are set. The proxy then reads the rest of the clients 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 disallowed, any cookie responses are deleted.
- If SSL is allowed and the connection uses SSL, the connection continues. No further processing can take place, since 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 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 clients 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 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 `(Cant connect to host)` to the client and drops the connection.
- 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 is 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 has been passed to the client, the proxy closes the connections to the client and the server and terminates the thread.

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.

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, then 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 single or range IP addresses). If a match is found, the connection is closed with a response (455) refusing reception.

Having passed the peer-address check, the proxy thread next attempts the typical proxy rule match steps (given earlier), except that only the source address is checked. For each rule that matches, an SMTP connection is attempted to the MTAs listed as destination for the rule.

Spam Control

Unsolicited electronic mail is colloquially-known as `spam`. In SunScreen EFS 3.0, 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 non-server (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)

server suffix is simply an ASCII character string.

See the section below entitled “SMTP Proxy Operation” for details regarding how these restrictors are used.

`Spam` restrictors are defined using the command-line interface, and the `mail_spam` sub-command of `ssadm edit` in the administration GUI.

The following is an example of what you type to display the current set of spam restrictors while logged in to the primary Screen:

```
admin% ssadm -r primary edit Initial
edit> mail_spam list
“total-nonsense.org”
“0.0.0.0..255.255.255.255”
```

The above example listing shows two entries, one to refuse email from the server 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).

The following is an example of what you type to add an additional restriction while logged in to the primary Screen:

```
edit> mail_spam add complete-spam.net
edit> quit
```

The mail_spam restrictors are stored in the database used by the vars variables. It is not necessary to type save before quit like above if only authuser, proxyuser, logmacro, or vars entities have been altered. If you attempt to save without changing entities other than these types, you are reminded by a message:

```
edit> save

lock not held

failed (status 244)
```

This is a non-fatal message in this situation; you can simply quit the configuration editor at this point.

Once changes have been made to spam restrictors, the current policy must be (re)activated to install the new objects and to propagate these changes to secondary Screens.

The following is an example of what you type to remove a restriction while logged in to the primary Screen:

```
edit> mail_spam delete lite-spam.com
```

From the command line, 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 policy rules, allows you to restrict which destination domains are accepted by the proxy.

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 the section below entitled "SMTP Proxy Operation" for details regarding how these restrictors are used.

Relay restrictors are defined using the command-line interface, through the `mail_relay` sub-command 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"
```

The above (example) listing shows two entries, one to set a base domain to allow in recipients, the other to block a private sub-domain 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
```

Note – The `mail_relay` restrictors are really stored in the database used by the `vars` variables. It is not necessary to enter `save` before `quit` above if only `authuser`, `proxyuser`, `logmacro`, or `vars` entities have been altered. If you attempt to `save` without changing entities other than these types, you will be reminded by a message:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message in this situation; you can simply `quit` the configuration editor at this point.

Once changes have been made to relay restrictors, the current policy must be (re)activated to install the new objects and to propagate these changes to secondary Screens.

The following is an example of what you type to remove a restriction (while logged into the primary Screen), type:

```
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
```

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
```

Finally, the following is an example of what you 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 EFS 3.0 represents addresses (in numerical order by IP address), rules that refer to multiple destination addresses give rise to attempting 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 use 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 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 is designed to allow 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 EFS 3.0 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 server Name System 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 multiple Screen management features of SunScreen EFS 3.0 make 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 actually allow SMTP-based email to be served. A typical configuration often allows outside email to be filtered through the proxy to one or more inside mail transfer agents (MTAs) 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 entirely-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 actually store and forward the incoming email.

telnet Proxy

The telnet proxy provides a virtual terminal relay. It 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 / password. If an invalid user name / password are sent, the telnet proxy sends an error to the user and closes the connection. If the user name / password are valid, then the source and destination addresses are checked against the Screens 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(es) 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 (shown in the section, “Policy Rule Matching”) is employed. 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 is relayed (uninspected) by the thread in both directions until either end terminates.

Other telnet Proxy Issues

The telnet proxy utilizes 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-echos *username*/*hostname* input and single-echos the password.

Note – By configuring the telnet proxy on a Screen, the actual telnet service into that system becomes unavailable. To avoid confusion, define the destination address of proxy rules so as to exclude all of your Screen(s) addresses. (You can still use the `rlogin` protocol to access the Screen, as necessary, as well as being able to 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: 2.0
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`.

You can have the Screen decrypt incoming traffic from a client before passing it to the proxy. It requires two rules. The following is an example of using the telnet proxy with SunScreen SKIP:

```
edit> add rule telnet proxyclient localhost SKIP_VERSION_2 ...
edit> add rule telnet proxyclient 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 proxyclient proxyserver PROXY_TELNET
```

User Authentication

SunScreen EFS 3.0 provides the ability to configure user entities. These entities are used to authenticate administrators *of* Screens and to allow access *through* the Screen when using proxied services.

Authentication provides the ability to verify the identity of users. The user facilities of SunScreen EFS 3.0 employ both internal and external means of proving user identity. Further, user entities can be configured in various ways depending on the role or roles they are to play with respect to SunScreen EFS 3.0 administration and use.

SunScreen EFS 3.0 proxies provide a means to validate, regulate, and extend the abilities of certain services beyond those afforded by kernel-based stateful packet filtering.

SunScreen EFS 3.0 User Model

The SunScreen EFS 3.0 user model provides two levels of user identification. These are the Authorized User (abbreviated *authuser*) and the Proxy User (abbreviated *proxyuser*).

Authorized Users

The Authorized User is represented as a named common object. Each such object is intended to describe an individual human user distinct from all others. The attributes of an Authorized User provide a repository for demographic and authentication data about that individual.

Authorized User objects contain information sufficient to allow authentication of users of SunScreen EFS 3.0. Validation information can either be: (1) simple text password or (2) SecurID(R) token PASSCODE; users can also be configured to have both means of authentication.

Access to and use of the administrative functions of SunScreen EFS 3.0 require the establishment of the Authorized User identity before administration is allowed. Both the Login Screen of the administration GUI and the `login` sub-command of the `ssadm` command line facility reference an Authorized User object.

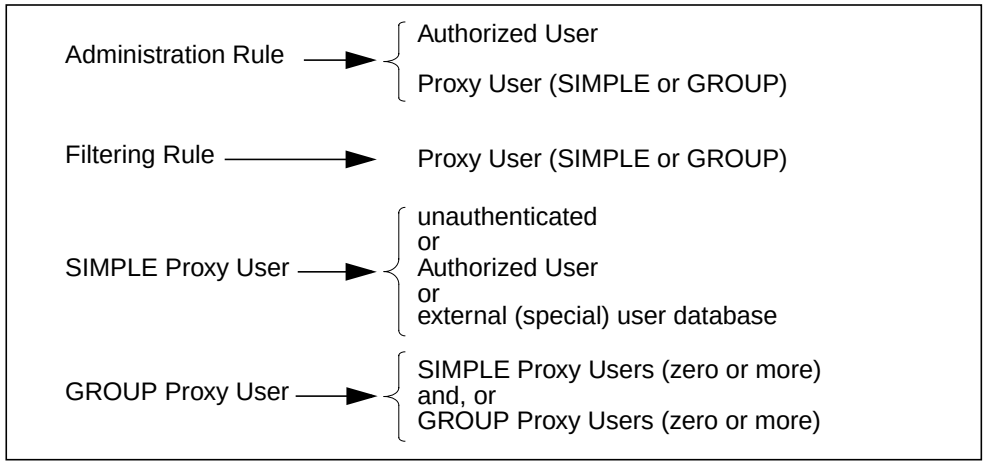
Authorized User authenticity establishes only the identity of a user, but does not itself describe any of the various roles a user can play in use of SunScreen EFS 3.0. Role establishment is afforded in one of two ways: (1) reference within the User field in the administrative access rules of a policy, (2) reference from a packet filtering rule that utilizes user authentication (proxies).

Proxy Users

The Proxy User is a named common object, and is distinct from the Authorized User. Proxy Users are either SIMPLE or GROUP objects. A SIMPLE object is used to provide for and establish an association between an individual human and a role that human plays in usage of the facilities controlled by SunScreen EFS 3.0. GROUP objects are used to allow creation of collections of SIMPLE Proxy Users that share common access to facilities; GROUPs streamline the task of allowing or removing access to established facilities.

Some special Proxy User objects also provide the means to *map* external collections of users into the access control facilities of SunScreen EFS 3.0. In SunScreen EFS 3.0, external access to SecurID(R) 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 following diagram summarizes the relationship between Rules, Authorized Users, Proxy Users, and external user databases:



The names of Authorized Users and Proxy Users are distinct, and it is perfectly allowable to have objects with identical names in each. You should 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 (distinguished names, employee numbers, etc.) and Proxy Users by names that reflect their normal user login names deployed on server systems (for example: Unix login name). Names cannot contain any of the following characters:

“!”, “#”, “$”, “%”, “^”, “&”, “*”, “{“, “}”, “[“, “]”, “<“, “>”, “””, “', “?”, “‘”, “/”, “@”, or NUL characters.

Space, tab, and other *whitespace* characters are allowed in names, but in doing so you should be prepared to supply quotation marks in some situations in order to protect such whitespace within names.

Note – In examples, the names of Authorized Users, Proxy Users, and other user naming items are often deliberately chosen to be different for purposes of clarity and illustration.

Authorized User Object Definition

The Authorized User and Proxy User objects can be created and managed by both the administration GUI and the command line interface. The administration GUI pages that manipulate these objects have already been elaborated in the administration GUI chapter. 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` — 1-255 characters, not including enabled or disabled enablement flag for the entire object; if disabled, authentication of the associated user is always denied; 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* 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 sub-items:

"*passwd*"— the plain-text password string; should either be empty (for example, `""`) or contains a one to eight character password; if this field is non-empty, then the next sub-item (`crypt_password=`) should not occur.

`crypt_password="`*cryptpasswd*`"` — (optional) the encrypted version of the plain-text password string; if this sub-item is present, then the plain-text password string (above) should be empty *enabled* | *disabledenablement* flag for this simple-text password authentication method; if disabled, any password presented for authentication of this user is not compared against this sub-item; the default is *enabled*.

Note – The processing of *passwd* and `crypt_password=` sub-items 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 plain-text using the (non-empty) *passwd* sub-item. Thereafter (for example, whenever the object is edited), the `crypt_passwd=` sub-item can be used to retain a password without having to know (or retype) the plain-text 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 EFS 3.0 user descriptions without the SunScreen EFS 3.0 administrator needing to know the users plain-text passwords. This fact also means that the content of the SunScreen EFS 3.0 Authorized User database is maintained with file permissions that prevent access from all but *root* users of the SunScreen EFS 3.0.

The `securid=` item has the following sub-items:

"*securidname*" — User login name associated with this users SecurID token in the ACE/Server database enabled | disabled enablement flag for this SecurID authentication method; if disabled, any password presented for authentication of this user is not be 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 should be presented first.

Authorized User Object Creation

The Authorized User object is manipulated using the `authuser` sub-command of `ssadm edit`. `authuser` 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 sub-items as defined above.

`delete` "*name*"— deletes a named object

`print[,`*sortopt*`] ["`*name*`"]`— display one or more objects; if an object name is given, then only that objects definition is displayed; otherwise all Authorized User objects are displayed

sortopt can be:

- ascascending order by name (case-sensitive)
- descdescending order by name (case-sensitive)
- iascascending order by name (case-insensitive)
- idescdescending order by name (case-insensitive)
- raw order stored in database
- default is `asc`

`names [,`*sortopt*`]`— display 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
- default is *asc*.

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”
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”
```

Note – Although the output produced by `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.

Also, although `print` outputs all tag names in capital letters (for example, REAL_NAME=), these tags are recognized in a case-insensitive manner on input (for example, real_name= and REAL_NAME= are equivalent.)

Note – Because of the way in which passwords are encrypted, it is unlikely that any `add` operation will yield a particular `crypt_password=` encoding of any given plain-text password. In fact, there are 4096 different encryptions of any given plain-text password.

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 plain-text form.

An alternate means of (re)creating the above Authorized User object, while logged in to 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
```

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

Tip – It is not necessary to type `save` before `quit` above if only `authuser`, `proxyuser`, `logmacro`, or `vars` entities have been altered.

If you attempt to save without changing entities other than these types, you are reminded by a message:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message in this situation; you can simply `quit` the configuration editor at this point.

Note – See Chapter 3, "Administration Graphical User Interface Reference" for more information regarding which common objects do not require the use of `save`.

Once changes have been made to Authorized User objects, the system configuration must be (re)activated to install the new objects and to propagate these changes to secondary Screens.

In each of the above `add` operations, the (two) enablement items have been allowed to default to enabled.

The following is an example of what you type to re-create the above Authorized User object but causing it to be 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”
```

The following is an example of what you type 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”
```

The following is an example of what you type 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”
```

The following is an example of what you type 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"
```

The following is an example of what you type 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
"jeff.hogg"
"admin"
```

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, then, 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 sub-item; if 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.

Proxy User Object Definition

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 be used: (1) none, (2) an Authorized User object, (3) an external authentication mechanism.

The relationship between SIMPLE Proxy Users and authentication mechanism was shown in an illustration previously.

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 – In SunScreen EFS 3.0, 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 (1-255 characters).
- `enabled` | `disabled enablement` — flag for the entire object; if disabled, authentication of the associated user is always denied; default is `enabled`
- `group` | `simple type` — designator of the object; its almost always possible to omit this on input as 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 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 the name of an Authorized

User object to be used to authenticate this user role; if 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 users 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 it is permissible to `add` a GROUP Proxy User object, including a complete list of its members, special commands `addmember` and `deletemember` are provided to edit the membership list of a GROUP.

Proxy User Object Creation

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; 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; attempting to remove an unknown member is ignored.
- `print[,`*`sortopt`*`] ["`*`name`*`"]` — Display one or more objects; if an object name as given, then only that objects definition is displayed; otherwise, all Proxy User objects are displayed.

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
- default is `asc`

`names [,`*`sortopt`*`]` — display the names of all objects *sortopt* can be:

- *asc ascending* order by name (case-sensitive)
- *dess 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
- default is *asc.*

The following is an example of what you type 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”
```

The following is an example of what you type 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
```

The following is an example of what you type 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"
```

Note – This 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.

Next, is an example of what you type to add the members of the example GROUP:

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

Note – 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.

Note – It is not necessary to type `save` before `quit` above if only `authuser`, `proxyuser`, `logmacro`, or `vars` entities have been altered.

If you attempt to save without changing entities other than these types, you are reminded by a message:

```
edit> save
lock not held
failed (status 244)
```

This is a non-fatal message in this situation; you can simply quit the configuration editor at this point.

Note – See Chapter 3, “Administration Graphical User Interface Reference” for more information regarding which common objects do not require the use of save.

Once changes have been made to Proxy User objects, the system configuration must be (re)activated to install the new objects and to propagate these changes to secondary Screens.

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

The following is an example of what you type 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”
```

The following is an example of what you type 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"
```

The following is an example of what you type 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 by the processing logic upon them.

There are fundamentally two types of authentication processing performed in SunScreen EFS 3.0. 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 you are 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,” which follows.)

The first type of processing is the authentication of an Authorized User object directly. This form occurs when authenticating an administrator of SunScreen EFS 3.0, either through the administration GUI Welcome page or by the `ssadm login` sub-command. 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”).

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 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 pre-installed `anonymous` and `ftp` Proxy User objects are examples of this type of authentication.

Note – There is nothing special about the names `anonymous` and `ftp` for these pre-installed objects. You can create additional null authentication Proxy User objects and use them in-lieu of or in addition to the pre-installed 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").

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:

```
/extmethodname/backendusername
```

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

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

Note – There is nothing special about the names radius and securid in these pre-installed SPECIAL objects; they are distinguished by their special radius or securid items. You can create additional SPECIAL authentication methods and use them in-lieu of or in addition to the pre-installed ones. This allows the creation of 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. Interestingly, note that there are two means for utilizing SecurID tokens for authentication: one within the Authorized User object, the other

through SPECIAL external Proxy User object. 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 EFS 3.0 Authorized User object and a specific SecurID tokenholder. The SPECIAL external authentication mechanism allows, in essence, *any* user that SecurID authenticates to satisfy its rigor.

Thus, depending upon the security requirements of your site, you can choose the mechanism to employ. Notably, the ability to establish authenticity for purposes of SunScreen EFS 3.0 administration is never available to SPECIAL external authentication.)

User Access Control Processing Logic

As previously referenced, once an identity is supplied, the would-be user is authenticated. It is still required that this (now authentic) user be allowed to perform the operation presently being attempting. 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 EFS 3.0 administrative activities and usage of network facilities controlled by FTP and telnet proxies.

Access control logic regarding SunScreen EFS 3.0 administration is covered in various other sections of this and other manuals. However, 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 EFS 3.0 proxies that perform user authentication do so by requiring a Proxy User object to be referenced by their rules. 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 referenced by the rule; if so, then the associated rule matches. (See below for more details regarding proxy rule matching.)

RADIUS User Authentication Processing Details

The RADIUS protocol provides the ability to centralize user authentication databases for widespread uniform deployment throughout organizations. RADIUS was originally developed for use by terminal-server devices, but its open specification and reference implementation have allowed it to achieve a sort of universal joint

interconnection status in the world of external authentication mechanisms. 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:

- Security Dynamics ACE/Server Version 3.2 and 3.3
- Sun Directory Services (SDS)

The SDS gateway facility enables SunScreen EFS 3.0 to tap into authentication using LDAP user databases.

With respect to a RADIUS server, the Screen plays the role of a *client* host or, for our purposes, a RADIUS Requestor.

Note – The term *client* is quickly subordinated here, to avoid any confusion with clients of SunScreen EFS 3.0-*provided* services.)

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 server(s) 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 EFS 3.0 comes installed with nearly all parameters prefigured for immediate use. Four remaining configuration elements needing post-installation attention are:

- Address object definition for the RADIUS servers
- Access rules allowing RADIUS protocol access to those servers
- Configure the RADIUS Requestor to use the defined server Address object(s)
- Configure the RADIUS Requestor with SunScreen EFS 3.0s node secret

The SunScreen EFS 3.0 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 – It is suggested that a single Address group object be defined to collect all RADIUS servers for ease in creating server access rules.

To allow the RADIUS Requestor to function, the Screen must be configured to allow it access to the server(s) through the `radius` service Common Object, which comes pre-installed.

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 ... }          (name(s) of address object(s)
of RADIUS server(s))
description="descriptive text"      (optional)
enabled | disabled                  (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 global "*RADIUSServers*" variable. Another is to use "avoid SCREEN" attributes on Address objects and instead use the sys= item to create Screen-specific "*RADIUSServer*" variables. Of course, combinations are also possible. And, naturally, the logic prefers Screen-specific Address objects and variables, over global ones.

The Address object(s) (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 – Due to the way SunScreen EFS 3.0 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 sub-items 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(s). 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"          (node secret string)
description="descriptive text"(optional)
enabled | disabled          (default is enabled)
```

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

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

Once Addresses, Rules, and variables have been established, the configuration must be activated 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.

The following is an example of what you type 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 } { } ...
```

The following is an example of what you type to create a rule to allow RADIUS-Requestor-to-server access, while logged into the primary Screen:

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

The following is an example of what you type 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"
```

The following is an example of what you type 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
```

Save and activate the configuration:

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

For example, given a valid, RADIUS-hosted user `gooduse` 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.
```

Note – `lib/user_authenticate` is a low-level program used internally to perform user authentication; its interface is not supported for general use. It echoes the password as it is typed.

User Databases

User databases are registries for user entities that are private to the SunScreen EFS 3.0 system.

There are two types of entities: Authentic users and Proxy users.

- The authuser database is manipulated by the `authuser` command in the configuration editor.
- The proxyuser database is manipulated by the `proxyuser` command in the configuration editor.

Each type of entity has a string for its *name.*

The `authuser` and `proxyuser` namespaces are distinct entities.

Editing *command verbs* and *item names* are case-insensitive; however, documentation generally uses all-CAPS in examples for emphasis.

For example:

```
edit> authuser add Ron password={ raygun }
```

is identical to:

```
edit> authuser add Ron PassWorD={ raygun }
```

The `add` verb performs a complete overwrite of the named item, which is important in the case of `proxyuser` *GROUP* entities, where the `addmember` and `deletemember` verbs have been provided to manipulate member lists.

The output of the `print` verb is suitable for re-use through `add`. The `print` verb outputs a canonical form, including defaults for required items. Here again, you need to address `shell` escape issues.

Each type of entity has an enablement flag (*enabler*), while some sub-fields also have enablement flags:

`ENABLED`

-or-

`DISABLED`

Where the default is `ENABLED`

authuser Database

authuser is intended to correspond one-to-one with actual persons. The authuser entity contains parameters that are required to verify the authenticity of a given individual authuser.

Note – Entities do not reflect user roles; that is what the proxyuser entities are for.

---

authuser entities allow authentication by simple password or SecurID token *PASSCODE* (a given authuser can be configured for both types; satisfying either one is sufficient to cause authentication).

authuser entities provide for multiple simple passwords; however, the administration GUI is only prepared to deal with zero or one.

authuser entities have the following required items:

name — However, to be of any great use, at least one of the following authentication mechanisms must also be given:

PASSWORD={ *plain-text* [CRYPT_PASSWORD=*crypt-text*] *enabler* }

SECURID={ *securid-name enabler* }

If the CRYPT_PASSWORD item is given, the *plain-text* is given as "" (empty placeholder string).

Any PASSWORD item(s) precedes a SECURID item, and only one SECURID item is allowed or used (ACE/Server contains a mechanism to allow a given user to use multiple token devices).

authuser entities have the following optional items:

- *enabler*
- CONTACT_INFO=*contact-info-string*
- DESCRIPTION=*desc-string*
- REAL_NAME=*real-name-string*

The administration GUI only expects zero or one of each of the above, but the command-line tools do not restrict their numbers.

authuser passwords are stored in encrypted form (DES Unix /etc/passwd style, up to eight characters).

When the configuration editor is presented with a *plain-text* password, it translates it into an encrypted item, because it cannot display *plain-text* form.

The configuration editor stores the encrypted items directly and uses them during database build operations. When you do not know a user password, they can also be used to paste-in user passwords from `/etc/passwd`.

`authuser` example with simple password authentication:

```
edit> authuser add harry.bovis password={ secret }edit> authuser
print harry.bovis
"harry.bovis" ENABLED PASSWORD={ ""
CRYPT_PASSWORD="upO7711Q0CRaA" ENABLED }
```

Output of `print` is actually a single, long line.

Note – If you try this example, the password encryption logic will probably generate a different *crypt-text* string for the password "`secret`". There are 4096 different possible encryptions for any given *plain-text* password.

`authuser` example with SecurID authentication:

```
edit> authuser add larry.hovis securid={ lhovis }
edit> authuser print larry.hovis
"hovis" ENABLED SECURID={ "lhovis" ENABLED }
```

`authuser` example, with both simple password and SecurID authentication:

```
edit> authuser add ac-dc password={ rockU } securid={ acdc }
edit> authuser print ac-dc
"ac-dc" ENABLED PASSWORD={ "" CRYPT_PASSWORD="fOE25hj8FhvIM"
ENABLED } SECURID={ "acdc" ENABLED }
```

Output of `print` is actually a single, long line.

`proxyuser` is intended to provide the idea of user roles to control the inclusion of users within groups used by Screen rules (through the *GROUP* entity type); they provide a means of mapping authentic users into their `roles` (user identities) on backend server systems (through the `AUTH_USER_NAME` or `BACKEND_USER_NAME` items); and they provide inclusion of external user authentication means (through the RADIUS item).

proxyuser Database

proxyuser entities have the following required items:

- *name*

Including two basic types:

SIMPLE or *GROUP*

Often, you do not have to enter the type specifier, as the configuration editor usually infers the type from the other items presented.

proxyuser items that are allowed in (and imply) *SIMPLE*:

- AUTH_USER_NAME=*auth-user-name* binds the proxyuser to an authuser entity. This then implies the requirement to authenticate using the methods and passwords specified for the latter.
- BACKEND_USER_NAME=*backend-user-string* is the string passed to a backend server by proxies as the name of the user on that server. If this item is missing, the *name* of the entity is supplied to the backend server.
- RADIUS is a special entry that you generally do not type.

proxyuser items that are allowed in (and imply) a *GROUP*:

MEMBER_NAME=*proxy_user_name* is repeated for each member of a *GROUP*.

Note – MEMBER_NAME is not often used directly for the add verb, in favor of the addmember and deletemember verbs.

---

proxyuser items are allowed in either *SIMPLE* or *GROUP*:

DESCRIPTION=*desc-string*

proxyuser example — the anonymous entity:

```
edit> proxyuser add anonymous BACKEND_USER_NAME=anonymous
edit> proxyuser print anonymous
"anonymous" ENABLED SIMPLE BACKEND_USER_NAME="anonymous"
```

This entity requires no authentication in any rule where it is included. Access is allowed by mentioning the name anonymous.

Note – There is nothing special, within the SunScreen EFS 3.0 authentication subsystem at least, about the name anonymous; the FTP protocol assigns special meaning, but the authentication logic is unaware of this.

proxyuser items provide a means by which non-well-known but un-authenticated access can be granted.

For example:

```
edit> proxyuser add backdoor SIMPLE
```

By mentioning the user backdoor, access is granted.

proxyuser example, authenticated user entity:

```
edit> proxyuser add ronco AUTH_USER_NAME=ron.popeil
edit> proxyuser print ronco
"ronco" ENABLED SIMPLE AUTH_USER_NAME="ron.popeil"
```

The BACKEND_USER_NAME defaults to "ronco" when used.

proxyuser example, authenticated user entity with alternate backend name mapping:

```
edit> proxyuser add msd-root AUTH_USER_NAME=marc.dye
BACKEND_USER_NAME=root DESCRIPTION="map msd to root"
edit> proxyuser print msd-root
"msd-root" ENABLED SIMPLE AUTH_USER_NAME="marc.dye"
BACKEND_USER_NAME="root" DESCRIPTION="map msd to root"
```

Output of print is actually a single, long line.

The backend server supplies the user name "root" upon connection establishment.

proxyuser example, creation of a *GROUP*:

```
edit> proxyuser add crow SIMPLE
edit> proxyuser add servo AUTH_USER_NAME=tom.servo
edit> proxyuser add inbound-ftp-users GROUP DESCRIPTION="users
allowed inbound access as themselves"
edit> proxyuser addmember inbound-ftp-users crow
edit> proxyuser addmember inbound-ftp-users servo
edit> proxyuser print
"anonymous" ENABLED SIMPLE BACKEND_USER_NAME="anonymous"
"crow" ENABLED SIMPLE
"inbound-ftp-users" ENABLED GROUP MEMBER_NAME="crow"
MEMBER_NAME="servo" DESCRIPTION="users allowed inbound access
as themselves"
"radius" ENABLED RADIUS SIMPLE
"servo" ENABLED SIMPLE AUTH_USER_NAME="tom.servo"
```

An FTP proxy rule, which uses *GROUP* `inbound-ftp-users`, allows the user `crow` without a password, and the user `servo` with whatever authentication the `authuser` entity `tom.servo` specifies. In both instances, the name of the `proxyuser` entity matched is supplied to the backend FTP server.

proxyuser example, more complete, allowing RADIUS authentication and groups within groups:

```
edit> authuser add dilbert SECURID={ wiseguy }
DESCRIPTION="contract engineer"
edit> authuser add dogbert PASSWORD={ trusted }
DESCRIPTION="management consultant"
edit> authuser print
"dilbert" ENABLED SECURID={ "wiseguy" ENABLED }
DESCRIPTION="contract engineer"
"dogbert" ENABLED PASSWORD={ ""
CRYPT_PASSWORD="O6JeSCIwq0LvA" ENABLED }
DESCRIPTION="management consultant"
edit> proxyuser add dilbert AUTH_USER_NAME=dilbert
BACKEND_USER_NAME=guestengr1 DESCRIPTION="contractor mapped to a
guestengr"
edit> proxyuser add dogbert AUTH_USER_NAME=dogbert
edit> proxyuser add engg GROUP
edit> proxyuser addmember engg dilbert
edit> proxyuser addmember engg radius
edit> proxyuser add inbound GROUP
edit> proxyuser addmember inbound dogbert
edit> proxyuser addmember inbound engg
edit> proxyuser print
"dilbert" ENABLED SIMPLE AUTH_USER_NAME="dilbert"
BACKEND_USER_NAME="guestengr1" DESCRIPTION="contractor mapped
to a guestengr"
"dogbert" ENABLED SIMPLE AUTH_USER_NAME="dogbert"
"engg" ENABLED GROUP MEMBER_NAME="dilbert"
MEMBER_NAME="radius"
"inbound" ENABLED GROUP MEMBER_NAME="dogbert"
MEMBER_NAME="engg"
"radius" ENABLED RADIUS SIMPLE
```

Examine the way in which the FTP proxy allows access for “inbound”:

```
user prompt: dilbert@server
user passwd: sidpass@srvpass
```

Assuming *sidpass* validates “wiseguy” with SecurID:

Proxy supplies the user “guestengr1” password *srvpass*

```
user prompt: /radius/msd@server
user passwd: radpass@srvpass
```

Assuming *sidpass* validates "msd" with RADIUS:

Proxy supplies the user "msd" password *srvpass*

```
user prompt: dogbert@server
user passwd: trusted@srvpass
```

Proxy supplies the user "dogbert" password *srvpass*

References to non-existent user entities behave as if the entity existed but was DISABLED (virtual); DISABLED entities (either real or virtual) always behave so as to cause a non-terminal failure of processing.

For example:

- If a rule references a proxyuser that does not exist or is DISABLED, that rule is never matched.
- If a proxyuser group references a non-existent proxyuser entity, it behaves as if that reference did not exist.
- If a proxyuser references a non-existent authuser entity, it behaves as if it existed but was DISABLED.

Authentication logic used within the proxies is performed with a three-step check for a given user *username* and password *passstrings*:

1. If *username* begins with a method prefix (for example. /radius/), then that method is invoked, returning SUCCESS or FAILURE. Otherwise, *username* is used as the *pname* sought in the next check (2).

2. An ENABLED proxyuser *pname* is sought; if it exists. Then, if that proxyuser contains no AUTH_USER_NAME item, then authentication SUCCEEDS immediately (*pass* is not checked); otherwise, the value of that item is used as the *aname* sought in the next check (3), otherwise, *pname* is used as the *aname* sought in the next check (3),

3. An ENABLED authuser *aname* is sought; if it exists. Then if *pass* authenticates it, then authentication SUCCEEDS, otherwise authentication FAILS.

Currently, authentication logic used for Screen administrative access control uses only the third (3) stage of the logic given above (that is, no external method nor proxyuser matching is performed).

The user-portion of the rule-matching logic used by the proxies performs the three-step authentication given above. If that authentication fails, no rule match is attempted; otherwise, the is-member portion of the rule match is performed. Each rule references a proxyuser user *ruser*; *ruser* can be a *GROUP* or a *SIMPLE* user.

If *username* begins with a `method` prefix (for example, `/radius/`), then if *ruser* has an `ENABLED` member with `method` (for example, RADIUS) then `is-member SUCCEEDS`; otherwise, `is-member FAILS`. Otherwise, if *ruser* has an ENABLED member named *username*, then `is-member SUCCEEDS`; otherwise, `is-member FAILS`.

RADIUS Authentication

RADIUS uses a User Datagram Protocol (UDP)-based query and response protocol to authenticate users from databases maintained on remote servers.

There are a variety of server offerings from different commercial and public-server suppliers. Many public-server and some commercial servers are based on a reference implemention created by *Livingston*.

There have been a few iterations of the RADIUS protocol; the SunScreen EFS 3.0 iteration is based on the latest iteration, per RFC2138.

One of the major distinctions between the iterations is the UDP port number used.

RADIUS authentication requires the following:

- At least one routing-mode port on the Screen (requires an IP address and a UDP stack).

Minimum configuration on the Screen requires:

- Configuration of one or more RADIUS servers to query
- Configuration of a *node secret* used by the protocol to encrypt and authenticate to and from the server(s).
- Creation of Rule(s) that allow the Screen to contact the RADIUS server(s)
- Creation of `proxyuser` entities (as needed) to reference the `radius proxyuser` entity
- Creation of Rule(s) (as needed) that utilize the `proxyuser` entities utilizing RADIUS (created above)

`RADIUS var prg=auth name=`*`RADIUSServers`*

Note – The `vars` command in the configuration editor manipulates variables used for RADIUS configuration.

A var controls the RADIUS server(s) to query; its key is:

`[ sys=`*`scrnname`*` ] prg=auth name=`*`RADIUSServers`*

The pre-installed value for this `var` is:

```
# ssadm vars print prg=auth name=RADIUSServers PRG="auth"
NAME="RADIUSServers" DISABLED VALUES={ host="server1"
host="server2" host="1.2.3.4" } DESCRIPTION="RADIUS server
name(s) address(es) to query"
```

Note – var is initially DISABLED.

The *RADIUSServers* var is a multiple-valued one (uses the VALUES={ *item*)

VALUES are the names or the IP addresses of the RADIUS servers; names given are resolved from the Screen registry (not from DNS or NIS).

Up to eight server addresses can be given (or implied by the name(s) referenced).

The following is an example of what you type to create an Address for the RADIUS servers and to edit the var to refer to the proper servers and enable it by typing:

```
# ssadm edit config
edit> add address mork HOST 8.8.8.8
edit> add address mindy HOST 8.8.9.9
edit> add address radius-servers GROUP { mork mindy }
edit> vars add prg=auth name=RADIUSServers VALUES={
host=radius-servers } DESCRIPTION="RADIUS server name(s)
address(es) to query"
edit> save
```

Activate these changes.

In centralized management group Screen configurations, there are two possible means to control the use of different servers for different Screens.

The following is an example of what you type to create an Address (or GROUP) that has a standard name but that has the SCREEN attribute, as needed, for different Screens:

```
# ssadm edit config
edit> list screen “msd-screen" ... "hq-screen" ...
edit> add address bob HOST 199.190.177.6
edit> add address carol HOST 199.190.177.9
edit> add address ted HOST 134.70.254.11
edit> add address alice HOST 134.70.254.111
edit> add address radius-servers GROUP { bob carol } SCREEN
hq-screen
edit> add address radius-servers GROUP { ted alice } SCREEN
msd-screen
edit> vars add prg=auth name=RADIUSServers VALUES={
host=radius-servers }
edit> save
```

The following is an example of what you type to create multiple var *RADIUSServers* items, each with a sys=*Screen* keying value:

```
# ssadm edit config
edit> list screen "hq-screen" ... "msd-screen" ...
edit> add address bob HOST 199.190.177.6
edit> add address carol HOST 199.190.177.9
edit> add address ted HOST 134.70.254.1
edit> add address alice HOST 134.70.254.11
edit> add address hq-radius-servers GROUP { bob carol }
edit> add address msd-radius-servers GROUP { ted alice }
edit> ...
edit> vars add sys=hq-screen prg=auth name=RADIUSServers
VALUES={ host=hq-radius-servers }
edit> vars add sys=msd-screen prg=auth name=RADIUSServers
VALUES={ host=msd-radius-servers }
```

Combinations of these approaches are possible also.

RADIUS var sys=*scrn* prg=auth name=*RADIUSNodeSecret*

A var controls the node secret used by the RADIUS protocol to secure traffic to and from the server(s); its key is:

[sys=*scrnname*] prg=auth name=*RADIUSNodeSecret*

The pre-installed value for this `var` is:

```
edit> vars print prg=auth name=RADIUSNodeSecret PRG="auth"
NAME="RADIUSNodeSecret" DISABLED
VALUE="....-....1....-....2....-....3.." DESCRIPTION="shared
secret for (this) RADIUS client"
```

Note – var is initially DISABLED.

Each client has a node secret; this secret is also called the *shared secret* by the RFC; the servers maintain a list of the secrets of the clients for which they perform authentication services.

The RFC indicates that the node secret can be up to 32-characters in length (not including a null-terminator).

Note – Although the RFC specifies a 32-character maximum, many servers (especially those derived from the reference implementation) will only properly utilize a node secret of 16 characters or less in length.

The following is an example of what you type to create a var for each Screens node secret:

```
edit> vars add sys=hq-screen prg=auth name=RADIUSNodeSecret
VALUE="managed2death" DESCRIPTION="shared secret for (this)
RADIUS client"
edit> vars add sys=msd-screen prg=auth name=RADIUSNodeSecret
VALUE="++stayNalive++" DESCRIPTION="shared secret for (this)
RADIUS client"
```

Activate these changes.

Note – In multiple Screen configurations, you can decide to *cheat* and utilize a common node secret for all or many Screens. Should such a choice be made, the common var for the node secret (the pre-installed one, that is initially DISABLED) can be ENABLED and provided with a proper secret string.

The following is an example of what you type to utilize a common node secret for all or many Screens:

```
edit> vars add prg=auth name=RADIUSNodeSecret
VALUE="AlwAysAnAgrAms" DESCRIPTION="shared secret for all RADIUS
client screens"
```

This (global, default) secret is used if no specific one is configured for a particular Screen.

RADIUS `proxyuser` References

RADIUS authentication is enabled by including member references to the pre-installed `proxyuser` entity `"radius"` in GROUP `proxyuser`, where `radius` is to be trusted.

The following is an example of what you type to enable authentication:

```
edit> proxyuser add grp-w GROUP DESCRIPTION="w/ RADIUS"
edit> proxyuser add grp-wo GROUP DESCRIPTION="w/o RADIUS"
edit> proxyuser addmember grp-wo user1
edit> proxyuser addmember grp-wo user2
edit> proxyuser addmember grp-w grp-wo
edit> proxyuser addmember grp-w radius
```

Rules that reference (either directly or indirectly) `"grp-wo"` only allows users `"user1"` and `"user2"`; whereas, ones that reference `"grp-w"` allow those two users, plus any user that the configured RADIUS server(s) authenticate.

RADIUS Client—>Server Rules

Access from the routing-mode Screen to the RADIUS server(s) must be allowed due to the configuration steps above, a usable `Address` already exists; if it does not, create such a rule that allows the Screen to communicate with its server(s).

The following is an example of what you type to create a rule to allows the Screen to communicate with its server(s):

```
# ssadm edit config
edit> add rule radius localhost radius-servers ALLOW ...
```

Where the above assumes un-secured RADIUS protocol access; the RADIUS protocol employs a modicum of cryptography, but if SunScreen SKIP security is also needed, then the preceding simple rule does not suffice.

Other `vars` for RADIUS Configuration

The following additional variables are pre-installed and used to control the RADIUS client protocol; they are pre-ENABLED and generally need not be altered:.

```
edit> vars print prg=auth PRG="auth" NAME="RADIUSHolddown"
ENABLED VALUE="300" DESCRIPTION="seconds to ignore a
non-responsive RADIUS server"
```

Where the client logic avoids contacting an unresponsive server for *this* many seconds:.

```
PRG="auth" NAME="RADIUSRetryPasses" ENABLED VALUE="3"
DESCRIPTION="how many times to try each RADIUS server"
```

Where the client logic makes *this* many passes through the server list before giving up.

```
PRG="auth" NAME="RADIUSService" ENABLED VALUE="radius"
DESCRIPTION="RADIUS service / port # at which to query
server(s)"
```

Where the name of the RADIUS server port, as given in the Service registry.

```
PRG="auth" NAME="RADIUSTimeout" ENABLED VALUE="5"
DESCRIPTION="seconds to await each RADIUS server response"
```

The amount of time to wait for each response before sending another attempt.

The client logic attempts to contact only servers that have not been held-down during the first pass; subsequent passes contact each server regardless of driving record 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:

```
# servers TIMES (#passes + 1) TIMES timeout
```

This is an upper-bound because of the way the first pass avoids recently un-responsive servers; a lower-bound would be:

```
# servers TIMES (#passes - 1) TIMES timeout
```

So, for example, with two servers configured and the default values, the overall failure timeout would be less than: 2 x (3+1) x 5 = 40 seconds, and greater than: 2 x (3-1) x 5 = 20 seconds

Other RADIUS Protocol Items

The client implementation only attempts to use authentication; it does not ask the server to store any accounting information. The client implementation allows for a single node secret for each Screen. It is theoretically possible to use a distinct node secret for each server; however, this was foregone in favor of configuration simplicity whose resistance to basic limitations in the protocol was traded-off against memory size and computation time.

RADIUS Testing

The client implementation has been tested against a commercial server implementation provided with BSDIs BSD/OS, Version 3.1. This server appears to be derived from the *Livingston* reference implementation and has been tested against a port of the *Livingston* reference to Solaris.

Testing is done against Sun Directory Services (SDS) and ACE/Server 3.2 SecurID authentication.

Direct SecurID authentication requires the following:

- At least one routing port on the Screen, which requires an IP address and the TCP and UDP stacks.

Command Access to the Screen

Minimum installation is required for any SecurID authentication; this includes usage of SecurID for authenticating Screen administrative users:

- Creation of Rule(s) that allow primary and secondary ACE/Servers to communicate (if needed).
- Creation of Rule(s) that allow the routing-mode Screen to contact the ACE/Server(s).
- A functioning ACE/Server v3.0.1 or greater.
- SecurID Client Installed on the Routing-Mode Screen.
- Creation of Rule(s) that allow SecurID token holders hosts to contact the PIN server.
- Creation of `authuser` entities that use SecurID authentication.
- An additional configuration is required to utilize SecurID authentication for users of the proxies.
- Creation of `proxyuser` entities (as needed) to reference `authuser` entities with SecurID authentication(created previously).
- Creation of Rule(s) (as needed) that utilize the `proxyuser` entities utilizing SecurID (created previously).
- SecurID can be used indirectly through RADIUS, but without the ability to do PIN establishment.

ACE/Server Rules

You may find it easier to perform Screen setup without SecurID first, as it adds complexities. Consider that the Screen may be in the path between the ACE/Server primary server and its secondary(ies). These things considered, it is important to create rules that allow the ACE/Servers to communicate with each other: Create an address group to contain the various ACE/Servers addresses. Create a rule that allows them to communicate with each other using the `securidprop` service.

The following is an example of what you type to create an address group to contain the various ACE/Servers addresses, then create a rule that allows them to communicate with each other using the `securidprop` service:

```
# ssadm edit config

edit> add rule securidprop ace-servers ace-servers ALLOW ...
```

The above rule assumes un-secured communication. The ACE/Server protocol uses its own encryption, but if SunScreen SKIP security is also needed, the above simple rule does not suffice.

If the Screen is in the path between the primary and secondary(ies), you can manipulate rules that can be used to test primary —> secondary fallback.

SunScreen EFS 3.0 ACE/Client Rules

You can allow the ACE/Clients to contact the ACE/Servers to perform the SecurID authentication when configuring ACE/Server rules.

The following is an example of what you type to create another address group to contain the various ACE/Client and ACE/Agent hosts addresses, then create a rule that allows them to communicate with the ACE/Server(s) in the address group created above:

```
# ssadm edit config
edit> add rule securid ace-clients ace-servers ALLOW ...
```

Again, the above assumes un-secured communication and relies on the variety of forms of encryption within the ACE/Client protocol itself; but if SunScreen SKIP security is also needed, the above simple rule will not suffice.

ACE/Server Setup

Refer to the ACE/Server documentation for the setup required to get a server running.

As part of the server-side configuration, the routing-mode Screen must be configured as a Unix client on the server.

Ensure that you either add alias addresses as needed for all routing-mode interfaces or be certain that the address configured for the routing-mode Screen is the one from which SecurID requests originate toward the ACE/Server.

The result of the server-side setup is an important file named "`sdconf.rec`." This file contains security information required by the client to enable it to contact the server and to establish communication. Arrange for this file to be available on your system for the client-side setup.

Although it is possible to install ACE/Server on a multi-homed host, such as most SunScreen EFS 3.0 Screens, it is unfortunately not convenient to protect the ACE/Server through direct routing-mode installation.

SecurID Client Setup on a Routing-Mode Screen

To learn about the Screen as a client, see `sdconf.rec` from the server *after* it has been configured.

Two possible client mechanisms:

- Install ACE/Agent 3.3 on Solaris 2.6 SPARC.

  This software is on the CD with the server. Install it per the documented instructions.

- Install ACE/Client stub.

  The stub files are installed by the `SUNWicgSS` package:

  ```
  /opt/SUNWicg/SunScreen/etc/securid_stubclient_setup
  /opt/SUNWicg/SunScreen/etc/securid_stubclient.tar
  ```

Note – The first file is a script that uses the second file.

Become root in the directory where `sdconf.rec` resides, execute:

```
# /opt/SUNWicg/SunScreen/etc/securid_stubclient_setup sdconf.rec
```

Once the client is installed, be sure to test it.

The first time a new client contacts the ACE/Server, it receives the node secret to allow this first-time exchange.

Note – The first authentication request must be performed as `root`.

You also need a SecurID token that is configured for a user and for which user is activated on the routing-mode client. A simple means to test it out is to use the SecurID PIN Server:

```
# telnet localhost 3855
SunScreen Vx.0 SecurID PIN and Re-Keying Server
```

Type the SecurID login: *user*
Type PASSCODE: *passcode*

Whether the passcode is accepted or access is denied, the client and server have exchanged the node secret.

Note – If the above interaction receives an error about not being able to establish server communications, ensure that you used the correct login and so forth.

SecurID PIN Server Rules

The procedure SecurID client setup on a routing-mode Screen, leads to the issue of the SecurID PIN Server (when operating in daemon mode, `securid` functions as a SecurID PIN Server).

The PIN Server is used to allow SecurID token holders to enter into a dialog with the ACE/Server to establish the users PIN.

PIN establishment varies depending on the type of token device options selected on the server, and so forth. They all have in common the need to have a more extensive dialog with the server than a simple password request or response, or even a password, challenge, or response. This dialog is called *PIN Dance*.

SecurIDs ACE/Client and ACE/Agent software contain programs that contain PIN Dance. These programs are installed in place of normal user login programs and other security hooks on a Unix client system, and enables normal user account protection through SecurID.

The PIN Server is accessed from any reasonable telnet client program (the client must allow connection to an arbitrary server port).

Create Rule(s) to allow access to this server from all hosts, where SecurID token holders are able to perform PIN establishment.

There are security issues to be considered when deciding which hosts should be allowed:

- Unless SunScreen SKIP encryption is employed, the PIN Dance is performed in-the-clear. During the dialog, the PIN is transmitted from the client host to the server.
- PIN transmittal, except for the PINPAD-style token, allows each successful authentication using the other tokens to contain the PIN (as the first part of the PASSCODE).
- Since PASSCODEs are normally transmitted in-the-clear, disclosure of the PIN during the PIN Dance is not a problem. Again, except for PINPAD tokens, where the lack of PIN transmittal is a key feature, the SecurID system is quite susceptible to denial-of-service attacks. By ill-considered access to the PIN daemon, an attacker can keep tokens perpetually disabled, and from which they know the associated users name.

Once you have considered the above issues, the following is an example of what you type to create an address group to contain the various client hosts from which to allow PIN establishment, and what you type to create a rule that allows them to communicate with the Screen:

```
# ssadm edit config
edit> add rule "SecurID PIN" PIN-clients localhost ALLOW ...
```

SecurID `authuser` Entities

One or more `authuser` entities must be configured to use SecurID. The following is an example of what you type to edit the default Screen administrative user to use SecurID authentication:

```
edit> authuser print admin
"admin" ENABLED PASSWORD={ "" CRYPT_PASSWORD="1hp1R.xm.w63Q"
ENABLED } DESCRIPTION="(created by install)"
REAL_NAME="SunScreen Administrator (before)
edit> authuser add admin PASSWORD={ ""
CRYPT_PASSWORD="1hp1R.xm.w63Q" } SECURID={ screenadm }
DESCRIPTION="(created by install)" REAL_NAME="SunScreen
Administrator
```

Where `screenadm` is the login name by the token as known to the ACE/Server. This string is not interpreted by the Screen logic in any way.

Note – Configuring the Screen to use SecurID administration provides another way for you to, for example, inadvertently make a rule change that breaks SecurID authentication. Therefore, always have at least one *last ditch* simple password administration account configured.

SecurID now functions on the routing-mode Screen. By creating additional `authuser`s that employ SecurID authentication, mapping them using `proxyuser` entities, and referring to those `proxyuser`s in proxy rules, SecurID is used to authenticate users of the proxies as well.

SecurID User Authentication Processing Details

SecurID® is a one-time password mechanism supplied by Security Dynamics Technologies, Inc. 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 Security Dynamics offering before v3.2 used the former name, while v3.2 and thereafter use the latter (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 v3.x system with a version greater than or equal to it (client version <= server version).

SunScreen EFS 3.0 is compatible with ACE/Server v3.0.1 and greater.

The SunScreen EFS 3.0 product *does not* include the ACE/Server product, which must be purchased separately.

Typical SecurID authentication involves a hardware device (token) that generates a pseudo-random 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 pseudo-random value as well as a user's PIN are (idealistically) 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 EFS 3.0, you are expected to understand the ACE/Agent and ACE/Server implementation to a level sufficient to install and configure the SunScreen EFS 3.0 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 EFS 3.0 Stub Client

The Security Dynamics ACE/Agent software offering is only supported on SPARC versions of Solaris through version 2.6 (SunOS 5.6). Yet, SunScreen EFS 3.0 is supported on Solaris 2.6 and beyond, and on both SPARC and x86 platforms. To complete the SunScreen EFS 3.0 support matrix, Sun has developed a *stub client* installation mechanism.

The stub client allows SunScreen EFS 3.0 to be configured with a *minimum* of information such that it can communicate with an ACE/Server for purposes of authenticating users of SunScreen EFS 3.0-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 Security Dynamics.

In summary, for SecurID support for SunScreen EFS 3.0, if you are installing SunScreen EFS 3.0 on SPARC-based Solaris 2.6, you can choose either the stub client or the complete ACE/Agent installation on the Screen. For SunScreen EFS 3.0 on other platforms or versions of Solaris, you *must* use the stub client.

The installation of ACE/Agent can be performed prior to or after the installation of SunScreen EFS 3.0. The SecurID stub client configuration step can be performed any time after SunScreen EFS 3.0 installation. SunScreen EFS 3.0 does not require SecurID to function, so it is possible (even recommended) to 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 EFS 3.0 and its usage of SecurID authentication, it is necessary that the SecurID client software be installed on any Screen(s) that makes use of 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 to be used for authentication of SunScreen EFS 3.0 administrators, then the client software must be installed on all Screens. It is not necessary to install SecurID software on the SunScreen EFS 3.0 Administration Station platform (for remote administration), nor on the end-systems of users of SunScreen EFS 3.0-protected resources (for example, proxy clients or backend servers).

The installation of ACE/Agent is not discussed herein, as it is detailed fully in the documentation for that product. One important note regarding ACE/Agent use on SunScreen EFS 3.0 is that it is *not* necessary to actually *create* Solaris user accounts on the Screens that are protected by ACE/Agent login mechanisms to enable the authentication of SunScreen EFS 3.0 users by that Screen. (It is certainly permissible and recommended to use ACE/Agent authentication to secure the Solaris platform of a SunScreen EFS 3.0 system in any way deemed important for administration of that system as a Solaris platform; but it is not required to make any changes to the Solaris user configuration to make full use of SecurID within SunScreen EFS 3.0 itself.)

With those notes, all other issues regarding use of SecurID within SunScreen EFS 3.0 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 EFS 3.0 packages are added; they are:

- `/opt/SUNWicg/SunScreen/lib/securid_stubclient_setup`
- `/opt/SUNWicg/SunScreen/lib/securid_stubclient.tar`

In addition, the stub client installation requires a file that is created on the ACE/Server; this file is called:

- `sdconf.rec`

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 (primary and secondary) as well as cryptographic data that enables the SecurID client to establish secure and authentic communication with the ACE/Server.

When the ACE/Server administrator creates `sdconf.rec`, you must first inform the server of the SunScreen EFS 3.0 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; it is important that *all* IP addresses the Screen will use to access the ACE/Server 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 SunScreen EFS 3.0 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:

```
# /opt/SUNWicg/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 server(s) 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 EFS 3.0 is found in two places:

- `/etc/inet/services`
- SunScreen EFS 3.0 service object in the active configuration

The protocol is named `securid` in both locations.

In more robust SecurID configurations, a secondaryACE/Server is configured to serve as a backup to the primary 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 EFS 3.0 policy must contain rules that allow the SecurID client software on the SunScreen EFS 3.0 to access the primary and secondary ACE/Servers. This involves:

- Address object definition for the ACE/Server(s)
- Rules to allow the `securid` protocol access to the ACE/Server(s)

Tip – It is suggested that a single Address Group object be defined to collect both primary and secondary ACE/Servers for ease in creating server access rules.

For the primary and secondary 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; again, this can be altered 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 primary and secondary servers to communicate using `securidprop`.

SecurID PIN Establishment

Part of the use of SecurID tokens involves the establishment of the personal identification number or PIN. There are a number of variations possible regarding PIN establishment; 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 allows the possibility that the token-holder can establish their own PIN. The experienced SecurID user knows that the standard ACE/Agent client software allows token-holder PIN establishment using the shell surrogate program `sdshell`. SunScreen EFS 3.0 does not require the use of the shell surrogate to use SecurID authentication; this avoids the severe security problems and administrative difficulties that would be associated with creation of user accounts on the Screen for each token-holder. Each token-holder must nevertheless be able to establish their PIN.

The SunScreen EFS 3.0 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`
- SunScreen EFS 3.0 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 V3.0 SecurID PIN / Re-keying Serve
Enter SecurID login: loginname
Enter PASSCODE: passcode
```

The interaction is familiar to users of the `sdshell` 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 feel it is appropriate to allow PIN establishment. For example, you may wish to require PIN establishment to occur only from hosts *behind* your Screens and from external hosts whose traffic is protected by SKIP encryption.

- Rules to allow PIN establishment through the PIN server

Note – Some SecurID installations may not allow token-holders to do PIN establishment, opting instead for use of PINs which 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 EFS 3.0 policy utilizing SecurID authentication.

The example presumes the following pre-existent state:

- `screen` is our SunScreen EFS 3.0 Screen (as well as `localhost`)
- `admin` is our (remote) SunScreen EFS 3.0 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.

The following is an example of what you type to perform the SecurID stub client configuration (while *root* in a shell on `screen`):

```
# cd /var/tmp
# /opt/SUNWicg/SunScreen/lib/securid_stubclient_setup sdconf.rec
```

The following is an example of what you type to create the registry address objects to describe the ACE/Servers (while logged in to the Screen):

```
admin% ssadm -r screen edit Initial
edit> add address acemaster HOST ....
edit> add address aceslave HOST ....
edit> add address aceservers GROUP { acemaster aceslave } { }
...
edit> save
```

The following is an example of what you type to continue adding the SecurID client-to-server policy rule:

```
edit> add rule securid localhost aceservers ALLOW
```

And to add the ACE/Server server-to-server policy rule:

```
edit> add rule securidprop aceservers aceservers ALLOW
```

And the PIN server policy rule (actually, two rules are shown being created, 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
```

Note – These rules should be placed early enough in the policy to preempt other conflicting (DENY or less-secure) rules.

Now, 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"
```

Save and activate the augmented policy:

```
edit> save
edit> quit
% ssadm -r screen activate Initial
```

Now, perform PIN establishment of the token (from the Administration Station):

```
% telnet screen 3855
Trying 1.2.3.4...
Connected to screen.
Escape character is '^]'.
SunScreen V3.0 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:
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 re-enter 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 EFS 3.0 requires the UDP and TCP protocols and further 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).

Caution should be exercised in the deployment of SecurID authentication for protection of SunScreen EFS 3.0 administration (and indeed any other critical SunScreen EFS 3.0-control facility where authentication is required). Because the ability to authenticate using SecurID requires use of policy rules in SunScreen EFS 3.0, it is possible that 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. This can result in an administrative lockout. As a precautionary measure, at least one SunScreen EFS 3.0 administrator should always be configured with ALL access and a simple-text password (perhaps in addition to SecurID).

ACE/Server v3.2 and newer can be hosted on SPARC Solaris 2.6. Because of the additional security features of SunScreen EFS 3.0, it is tempting to install ACE/Server *on* the SunScreen EFS 3.0. This is a perfectly acceptable deployment, but you are cautioned to understand *thoroughly* the ramifications of installing ACE/Server on a multi-homed host. There are numerous complexities to be dealt with on an on-going basis if your SunScreen EFS 3.0 does not have a *single* IP address that can service all queries from other SecurID software components. (See your ACE/Server installation documentation regarding "Multiple Server IP Addresses.")

HTTP proxy

The HTTP proxy only blocks certain types of outbound Web references and content and does not indicate usage to attempt to protect interior Web servers.

Using the HTTP proxy helps implement NAT because it originates all outbound connections to Web servers. It reuses a single IP address for multiple browser clients.

The HTTP proxy configuration is configured on a per-rule basis in the configuration editor to remove *cookies* (see also the following limitations section); to block Active-X content; to block or allow all Java content (see the following HTTP proxy limitations); and to block SSL content.

The HTTP proxy configuration is configured on a global basis in the configuration editor to verify the list of acceptable signatures and hash values on Java ARchive (JAR) content.

The following is an example of what you type to manage JAR hashes:

```
edit> jar_hash parameters...
```

Hashes can be added, deleted, listed, or renamed. They are named; you assign the names, which are strings, ephemerally. They are used only for purposes of reference to items when managing them using the `jar_hash` command.

Signatures can be added, deleted, listed, or renamed. They are named and you assign the names, which are strings, ephemerally. They are used only for purposes of reference to items when managing them using the `jar_sig` command.

Both the JAR hashes and signatures are actually stored in the `vars` database.

Note – Currently, there is no provision to create Screen-specific versions of these data items.

HTTP Proxy Limitations

The `COOKIE` restriction simply removes cookies in both directions, rather than refusing requests to set and get them within the HTTP protocol; this behavior often breaks access to pages that are viewable by simply refusing the cookies, which is how today's modern browsers work.

JAR validation facilities acquire the hash and signature values that are added to the configuration.

ACTIVEX filtering currently blocks certain types of Java content and does not support certain types of FTP downloads. It multiplexes and demultiplexes URL references that contain explicit port numbers (for example: `http://foo.com:12345/`), which is good because without the proxy, and lacking a proper `www` state engine in the kernel, you must either forgo access to such URLs or allow insecure `tcpall` outbound from all browser hosts.

The proxy provides the option to allow `tcpall` from the routing-mode Screen outbound, which, in conjunction with the HTTP proxy, affords access to such URLs through the proxy.

However, the HTTP proxy can also have untoward security problems by permitting client host browsers to gain outbound access to systems and ports for which you did not intend.

For example:

```
# ssadm edit config
edit> add address sensitive1 HOST 222.1.1.1
edit> add address sensitive2 HOST 222.1.2.2
edit> add address sensitive GROUP { sensitive1 sensitive2 }
edit> add address browser HOST 222.1.3.3
edit> add address router HOST 222.1.254.254
edit> add address qe0 RANGE 222.1.254.0 222.1.254.255
edit> add address qe1 RANGE 222.1.1.0 222.1.1.255
edit> add address qe2 RANGE 222.1.2.0 222.1.2.255
edit> add address qe3 RANGE 222.1.3.0 222.1.3.255
edit> add address outside GROUP { } { qe1 qe2 qe3 localhost
router }
edit> add rule sensitive-stuff sensitive sensitive ALLOW
edit> add rule telnet sensitive router ALLOW
edit> add rule telnet localhost router ALLOW
edit> add rule telnet localhost sensitive ALLOW
edit> add rule www browser "*" ALLOW PROXY_HTTP
```

It appears as if `sensitive` is well protected, but, in fact, the browser can cause the HTTP proxy to access the `telnet` service of both `router` and `sensitive`.

Including the following rule worsens the configuration situation:

```
edit> add rule "common services" localhost "*" ALLOW
```

Such a rule affords *browser* access to any TCP service in any rule with the source `localhost` (unless contrary DENY rules are created)

SMTP Proxy

The SMTP proxy provides a minimum of filtering on inbound SMTP mail traffic and does not handle outbound SMTP mail well. Although, it can provide some control over inbound mail with respect to requests for relaying services.

You can, however, consider it to be a first-line of defense against mail spam. Its filtering is restricted to information that is either obtained from the client host or the DNS.

The SMTP configuration can be configured to restrict relaying of inbound mail on a per-rule basis in the configuration editor by specifying a RELAY flag on a rule, all inbound mail is accepted even if a recipient address specifies relaying the absence of the flag (or specifying NO_RELAY) causes the inbound recipients to be checked against a list of allowed relay servers; the default for this list is simply the server of the routing-mode Screen whereon the proxy is running.

Managing the list of allowed relay servers, uses the following command:

```
edit> mail_relay parameters...
```

Allowed servers are maintained in a list of unnamed items that can be added, deleted, and listed.

Managing the list of spam servers, uses the following command:

```
edit> mail_spam parameters...
```

spam servers are maintained in a list of unnamed items that can be added, deleted, and listed.

Currently, there is no way to create Screen-specific versions of these data items.

SMTP Mail Handling

Mail is relayed from any source address specified in the source of a rule to the first destination in the rule that accepts the connection. The order for contacting actual inside SMTP servers (for destinations with multiple addresses) is IP-numerical order (lowest-numbered host first). If more precise ordering is needed, create multiple rules.

SMTP Proxy Filtering

When the relay flag is *not* set on a rule, each recipient mailbox name is checked against a list of allowed servers on which to perform relaying of inbound mail server name matching.

APPENDIX **A**

Migrating From Previous SunScreen Firewall Products

This appendix contains a table that compares the commands used in SunScreen EFS, Release 2.0, and SunScreen SPF-200 to the equivalent commands used in SunScreen EFS 3.0.

Note – Typing the `ssadm edit` command invokes the configuration editor, which gives you 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 EFS 3.0 Equivalent
`httpp`	Not used
`sas_logdump`	`ssadm logdump`
`sas_main`	Java GUI
`sas_registry`	Java GUI
`smtpp`	Not used
`spf_admin_install`	Not used
`spf_upgrade_skip`	Not used
`ss_access`	`edit> add accessremote` and `edit> add accesslocal`
`ss_action`	Not used
`ss_active_config`	`ssadm active`
`ss_address`	`edit> add address`

TABLE A-1 Command Compatibility Reference Table

Old Commands	SunScreen EFS 3.0 Equivalent
ss_admin_user	edit> authuser add
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 *Screen* ...	ssadm -r *Screen* ...
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 *policy*
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_status	ssadm ha status
ss_had	Not used
ss_ha_restart	Not used
ss_init_interfaces	Not used

TABLE A-1 Command Compatibility Reference Table

Old Commands	SunScreen EFS 3.0 Equivalent
ss_install	ss_install
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_proxyuser	edit> proxyuser add
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_spam_list	edit> mail_spam add
ss_stateengine	edit> list stateengine
ss_traffic_stats	ssadm traffic_stats
ss_sys_info	ssadm sys_info
statetables -f	ssadm lib/statetables -f
telnetp	Not used

APPENDIX **B**

Command Line Reference

This appendix describes the command line user interface for SunScreen EFS 3.0 and the options available for each command.

The top-level programs that you run directly from a Unix shell prompt or shell script are available if the `/opt/SUNWicg/SunScreen/bin` directory is included in your $PATH.

Other sections in this appendix describe the commands of the two major SunScreen EFS 3.0 subsystems that interpret commands: the `ssadm` command and the configuration editor. There is also a section on *unstable* commands that are obsolete or otherwise not guaranteed to exist in future releases of this product.

The following sections describe:

- What is the command line?
- Unix (`shell`) commands
- `ssadm` command
- Configuration editor
- Unstable commands
- Network monitoring and maintenance

Note – As the system administrator for SunScreen EFS 3.0, the command line is most appropriate if you are familiar with the Unix shell and SunScreen EFS 3.0 concepts and terms. If you are new to SunScreen EFS 3.0, use the administration graphical user interface (GUI) to enter network settings.

What Is the Command Line?

All the functionality of SunScreen EFS 3.0 that is available through the administration GUI is also available through a command. Administering your Screens through the command line can be useful when you want to manage one or more remote Screen or if you use more than one network address.

You can access a Screen using the command line from its own keyboard, when the Screen is being administered locally and requires that you have superuser (`root`) access; or you can access a Screen using the command line from an Administration Station, when the Screen is being administered remotely and requires that you use SKIP encryption and an Administration User name and password.

You maintain user-controlled data, such as common objects like Edit, and policy objects like Rules and NAT entries, using the `edit` command that is a sub-command of `ssadm`.

When you need to look at or change a policy in some way like Move or Delete, you invoke the configuration editor and enter a series of commands that end with `save` and `quit` requests.

Note – Be sure to save change commands like `add`, `del`, `rename`, `renamereference`, `insert`, `replace`, and `move`, before you `quit`. Run `save` just before the `quit` command to avoid accumulating too many policy versions.

You invoke the configuration editor with the `edit` command, which is a sub-command of `ssadm`, and the name of your policy, such as Initial. Once it is running, the prompt becomes: `edit>`.

For a locally administered Screen, type:

```
# ssadm edit policy_name
```

For a remotely administered Screen, type:

```
# ssadm -r Screen_name edit policy_name
```

Sub-Command man Pages

When running Solaris 7, if you are not able to read the man pages for the following ssadm sub-commands, use the workaround shown below.

ssadm-activate(1M), ssadm-active(1M), ssadm-algorithm(1M), ssadm-backup(1M), ssadm-debug_level(1M), ssadm-edit(1M), ssadm-ha(1M), ssadm-lock(1M), ssadm-log(1M), ssadm-logdump(1M), ssadm-login(1M), ssadm-logout(1M), ssadm-logstats(1M), ssadm-patch(1M), ssadm-policy(1M), ssadm-product(1M), ssadm-restore(1M), ssadm-spf2efs(1M), ssadm-sys_info(1M), ssadm-traffic_stats(1M)

To read the man pages, type the following:

```
# man -F ssadm-activate
```

Unix (shell) Commands

The following Unix (shell) commands are available at your shell prompt when `/opt/SUNWicg/SunScreen/bin` is included in your $PATH.

- `ss_install`
- `screenInstaller`
- `adminInstaller`
- `ssadm`
- `ss_client`

Note – Commands used by the `skiptool` GUI can be found in the *SunScreen SKIP 1.5 User's Guide.*

The following table lists the SunScreen EFS 3.0 Unix (`shell`) commands and their descriptions. Many of these commands duplicate administration GUI functions, while others provide a context for other commands.

TABLE B-1 SunScreen EFS 3.0 Unix (`shell`) Command Summary

Unix Command	Description
`ss_install`	Run the text-based utility for creating the Initial SunScreen configuration. When combined with `pkgadd`, it is equivalent to using the installation-wizard graphical user interface.
`screenInstaller`	Run the graphical user interface for installing the SunScreen EFS 3.0 software on the Screen and for setting up an initial policy.
`adminInstaller`	Run the graphical user interface for installing the SunScreen EFS 3.0 software on the Administration Station. It is a quick way to add packages for the remote Administration Station.
`ss_client`	Provide communication between an Administration Station and a Screen that is running an earlier SunScreen firewall product release. ss_client is provided only for the purpose of remotely administering such products using the SunScreen EFS 3.0 system as a remote Administration Station.
`ssadm`	Primary command-line tool for SunScreen EFS 3.0 administration. `ssadm` sub-commands perform various operations such as editing and activating a SunScreen configuration, and examining the status of a Screen.

ss_install Command

A text-based command-line utility run during SunScreen EFS 3.0 installation to create an initial configuration. ss_install, combined with pkgadd, is the command-line equivalent to the installation-wizard graphical user interface.

Usage: ss_install

ss_install interactively queries you with various configuration options, creates a configuration, stores it under the policy name “Initial”, and activates it.

After ss_install is complete, the Screen is ready to be administered using the administration GUI or the command-line configuration editor and other tools.

screenInstaller Command

Runs the installation wizard that installs the SunScreen EFS 3.0 software on the Screen and creates the Initial configuration.

Usage: ScreenInstaller

adminInstaller Command

Runs the installation wizard that installs the SunScreen EFS 3.0 software on the Administration Station. It is also a quick way to add packages for the remote Administration Station.

Usage: adminInstaller

ss_client Command

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 EFS 3.0 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 Command

ssadm is the primary command-line tool for SunScreen EFS 3.0 administration. ssadm has a number of sub-commands that perform various operations such as editing and activating a configuration, and examining the status of a Screen.

The Unix 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 to encrypt IP network communications passing between them. See the *SunScreen SKIP User's Guide* for more information regarding SKIP encryption.

Usage:

ssadm [-b] [-n] *sub-command [parameters...]*

ssadm [-b] [-n] -r *remotehost* [-F *ticketfile*] *sub-command [parameters...]*

Options:

-b — Allow binary data (instead of text) in standard input and output.

-n — Do not read any input from standard input.

-r *remotehost* — Access remote Screen using address or hostname *remotehost*.

-F *ticketfile* — Use authorization ticket stored in *ticketfile*.

The available ssadm sub-commands are each described in the *ssadm Sub-Command* section of this document.

The -b option normally is not needed since those 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.

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 on 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 called "Initial," you would type:

```
# ssadm activate Initial
```

where `ssadm` is the command you want to execute, `activate` is the name of the `ssadm` subcommand, and `Initial` is the name of the policy you want to activate.

The `ssadm` command resides in the `/opt/SUNWicg/SunScreen/bin` 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 "`Initial`" on a remote Screen called `SunScreen1`, you would type:

```
# ssadm -r SunScreen1 activate Initial
```

where `ssadm -r` indicates that you want to execute a command on a remote Screen called `SunScreen1`, `activate` is the name of the `ssadm` sub-command, and `Initial` is the name of the policy you want to activate.

Note – A local `ssadm` command can be turned into a remote `ssadm` command by adding `-r` *remote_screen_name* immediately after `ssadm`.

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 Sub-Commands

The following commands, which can be used as the sub-command argument to the ssadm command, are described in this section.

- activate
- active
- algorithm
- backup
- debug_level
- edit
- ha
- lock
- log
- logdump
- login
- logout
- logstats
- patch
- policy
- product
- restore
- sys_info
- traffic_stats

ssadm Sub-Command Summary

The following table lists the SunScreen EFS 3.0 ssadm sub-commands and their descriptions. Many ssadm sub-commands duplicate administration graphical users interface functions, while others provide a context for other sub-commands.

TABLE B¬2 SunScreen EFS 3.0 ssadm Sub-Command Summary

ssadm **Sub-command**	**Description**
activate	Activate a Screen policy.
active	List information about the currently active policy.
algorithm	List algorithms supported by SKIP.
backup	Write a SunScreen backup file to standard output.
debug_level	Set or clear the level of debugging output generated by a Screen.
edit	Run the SunScreen configuration editor. See *Configuration Editor Sub-Command Summary*.
ha	Configure the features of a High Availability (HA) Screen.
lock	Examine or remove the protection lock that the configuration editor places on a policy file.
log	Maintain the Screen log file.
logdump	Filter or display log records, as retrieved by ssadm log get.
login	Authenticate a user for administrative access through ssadm to a Screen from a remote Administration Station.
logmacro	Expands a SunScreen logmacro object.
logout	Terminate the session created by ssadm login.
logstats	Print information about the SunScreen log.
patch	Install patch, as needed.
policy	Create, delete, list, rename Screen policies.
product	Print single line of descriptive SunScreen EFS 3.0 use.
restore	Read a backup file from standard input.
sys_info	Print a description of running SunScreen software.
traffic_stats	Report summary information about the traffic flowing through the SunScreen, classified by interface.

`activate` Sub-Command

`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.

Syntax:

`ssadm activate [-n] [-l]` *policy*

Options:

`-n` — Do not actually make the configuration active, just verify that it is valid.

`-l` — Do not send the configuration to other Screens in the centralized management group, only activate 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.

`active` Sub-Command

`ssadm active` prints out a description of the configuration that is currently being *executed* by the Screen. When run with the `-x` option, the actual 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 Unix 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, it might be 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 saved policy contains a full set of common objects in addition to the policy rules. The `-x` option is different from a normal policy that contains only policy rules and is meant to be combined with the currently defined common objects.

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.

`algorithm` Sub-Command

`ssadm algorithm` lists the SKIP algorithms that are available for a specified algorithm type.

Usage:

`ssadm algorithm` *type* [*skipversion*]

where *type* must be one of “`key`”, “`data`”, “`mac`”, or “`compression`”. *skipversion*, if supplied, must be either “`SKIP_VERSION_1`” or “`SKIP_VERSION_2`”.

backup Sub-Command

`ssadm backup` writes a Screen backup file to standard output.

Usage:

`ssadm backup [-v] >` *file*

The backup file contains the complete configuration of SKIP, 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 secret keys) that must be stored and disposed of appropriately to protect the integrity of the Screen.

debug_level Sub-Command

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 Unix `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.

`edit` Sub-Command

`ssadm edit` runs the SunScreen EFS 3.0 configuration editor.

Usage:

`ssadm edit` *policy*

`ssadm edit` *policy* `<` *file*

`ssadm edit` *policy* `-c` *commandstring*

See the "Configuration Editor Sub-Command" section 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 Unix shell it usually has to be quoted with single or double quotes.

`ha` Sub-Command

`ssadm ha` performs operations on a Screen in a high availability (HA) cluster.

Usage:

`ssadm ha` *function parameters...*

Functions:

`status` — Display status of the HA cluster.

`active_mode` — Put the Screen in active mode.

`passive_mode` — Put the Screen in passive mode.

`init_primary` *interface* — Turn a standalone (non-HA) Screen into an HA primary Screen, thereby creating a new HA cluster containing one Screen. `interface` is the interface to be used for the HA heartbeat and synchronization. `primaryIP` is the IP address (on the HA network) of the primary machine in the cluster.

`init_secondary` *interface primaryIP* — Turn a standalone (non-HA) Screen into an HA secondary screen ready to join an existing HA cluster. Where *interface* is used for the HA heartbeat and synchronization, and *primaryIP* is the IP address (on the HA network) of the primary machine in the cluster.

`add_secondary` *secondaryIP* — Add an initialized HA secondary Screen (*see* `init_secondary` above) into an existing HA cluster. This command is executed on the primary Screen in the HA cluster. Where *secondaryIP* is the IP address (on the HA network) of the secondary machine to be added.

`lock` Sub-Command

`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
```

`log` Sub-Command

`ssadm log` retrieves and clears the SunScreen EFS 3.0 log.

Usage:

`ssadm log get` *filter_args...*

`ssadm log get_and_clear` *filter_args...*

`ssadm log clear`

`logdump` Sub-Command

`ssadm logdump` is used to filter or display log records, as retrieved by `ssadm log get`.

Usage:

`ssadm logdump` *parameters...*

login Sub-Command

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 multi-user Administration Stations, any other user can snoop the admin user and password using ps, then (because SKIP is enabled from that host) access the Screen(s) as that user.

It is inadvisable to have a general-use Solaris system act as a SunScreen EFS 3.0 Administration Station. Additionally, *never* use the ssadm login command on a Solaris system while other users are logged in.

Caution – 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.

logout Sub-Command

`ssadm logout` terminates the session created by `ssadm login`.

Usage:

`ssadm -r` *remotehost* `logout`

`ssadm logout` is only available with the `-r` *remotehost* option.

Note – You are encouraged to log out from the SunScreen EFS 3.0 administration GUI on the Administration Station upon completion of the administrative tasks. If you remain logged into the Administration Station without activity for a twenty-four hour period, the administration GUI stops responding.

---

logmacro Sub-Command

`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 plain-text description string `desc` to be associated with the object.

logstats Sub-Command

`ssadm logstats` prints information about the SunScreen EFS 3.0 log.

Usage:

```
ssadm logstats
```

`patch` Sub-Command

`ssadm patch` installs a patch, as needed.

Usage:

`ssadm patch <` *patchfile*

If a SunScreen EFS 3.0 software patch is needed, detailed instructions are provided with the patch.

`policy` Sub-Command

`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*

Options:

`-a` — Creates policies with the specified names. The newly created policy contains no rules and reference the currently defined common objects.

`-c` — Creates a policy named *newname* as a copy of the existing policy named *oldname*.

`-d` — Deletes the named policies. The specified *policies* can be either generic policy names, such as "`Initial`", or specific versions, such as "`Initial.3`". When a generic policy name is specified and the `-v` option is specified, `ssadm policy -d` deletes all of the versions of the policy. When a specific version is specified, only that version is deleted.

`-l` — Lists the named *policies* (or all policies available if no *policies* are given). The `-v` option also lists all of the saved versions of the *policies*.

`-r` — Renames the existing policy *oldname* as *newname*.

product Sub-Command

ssadm product prints out a single line of text describing the SunScreen product in use. For SunScreen EFS 3.0 this is simply "EFS".

Usage:

```
ssadm product
```

restore Sub-Command

ssadm restore reads a backup file from standard input. The backup file must have been created using the backup command.

Usage:

```
ssadm restore < file
```

spf2efs Sub-Command

ssadm spf2efs converts a set of configuration data saved from a SunScreen SPF Screen into SunScreen EFS 3.0 format.

Usage:

```
ssadm spf2efs < file
```

sys_info Sub-Command

ssadm sys_info prints a description of the running SunScreen software.

Usage:

```
ssadm sys_info
```

For example:

```
# ssadm sys_info
Product:                    SunScreen EFS 3.0
System Boot Time:           03/15/1999 03:51 PST
SunScreen Boot Time:        Mon Mar 15 03:51:56 PST 1999
Version:                    Release 3.0 Beta1, March 10
                            1999(v0310991418)
```

`traffic_stats` Sub-Command

`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 obsolete. They are only used for abnormal maintenance or customer support functions, or as a temporary work around 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/screeninfo Command`

`ssadm lib/statetables -f Command`

`ssadm lib/support Command`

`ssadm SKIP Commands`

`ssadm lib/screeninfo` Command

`ssadm lib/screeninfo` runs in sequence several of the functions of the `ssadm lib/support` 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 Service if you encounter problems with your Screen.

`ssadm lib/statetables -f` Command

`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.

This command is often useful after activating a modified policy which disallows some traffic which was previously allowed. Without running `ssadm lib/statetables -f`, you allow any previously existing connections to remain active even if the new policy does not allow them. By running `ssadm lib/statetables -f`, you cause all previously existing connections to be disconnected and the active policy will apply to any subsequent connections.

`ssadm lib/support` Command

`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 Enterprise Services: http://www.sun.com/service/contacting.

The major functions are:

TABLE B-2 Support Command Functions

Functions	Description
`config`	Bring over configuration files for the active policy
`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.

TABLE B-2 Support Command Functions

Functions	Description
`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

`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 SKIP software. They are primarily useful when a Screen is not physically accessible or does not allow logins 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 1.5 User's Guide* 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 following table lists the data types that compose the Data Model as maintained by the configuration editor (`ssadm edit`) and the `ssadm policy` command.

TABLE B-3 Configuration Editor Object Type Name Summary

Object Type Name	Storage	Access Method	Description
`address`	common	named	Describe addresses of network elements
`screen`	common	named	Describe Screen objects and their relationships
`state engine`	common (read-only)	named	Describe filtering capabilities of packet filter engine.
`service`	common	named	Define network services that can be filtered
`interface`	common	named	Describe network interfaces of a Screen.
`certificate`	common	named	Refer to certificate used for SKIP connections
`time`	common	named	Define time intervals for time-dependent rules
`authuser`	external	named	Describe users for administration and/or proxy access
`proxyuser`	external	named	Describe users for proxy access
`jar_hash`	external	named	Describe Java archive hash (for HTTP proxy applet filtering)
`jar_sig`	external	named	Describe Java archive signature (for HTTP proxy applet filtering)
`logmacro`			
`mail_relay`	external	named	Describe mail relays (for SMTP proxy mail filtering)

TABLE B-3 Configuration Editor Object Type Name Summary

Object Type Name	Storage	Access Method	Description
`mail_spam`	external	named	Describe spam domains (for SMTP proxy mail filtering)
`policy`	policy list	named	Create, delete, rename, or list the defined policies
`filter rule`	policy	ordered	Describe network traffic flow policy
`nat rule`	policy	ordered	Describe NAT translations (read-only)
`local access rule`	policy	ordered	Describe who can access the Screen for local administration and what they can do.
`remote access rule`	policy	ordered	Describe who can access the Screen for remote administration and what they can do.
`VPN gateway`	policy	ordered	Describe how VPN hosts are protected behind certificates and tunnels
`VPN`	policy	ordered	Virtual object representing a collection of VPN gateways

Object types marked as having “common” storage in the table are normally stored in the common objects registry that is not part of any particular 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, it is necessary to 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 the table are stored as part of a policy. Policy objects often refer to common objects and therefore can have different meaning 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 immediate, and persist even if the “`save`” command is not used.

Object types marked as having “policy list” storage in the table represents the names of the policies themselves. Minimal capabilities are provided by the configuration editor to manage the policy. 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 Commands

The `ssadm edit` commands are used when running the configuration editor, which is responsible for maintaining the SunScreen EFS 3.0 configuration database.

The following table lists the SunScreen EFS 3.0 configuration editor `ssadm edit` sub-commands and their descriptions. Many sub-commands duplicate administration GUI functions, while others provide a context for other sub-commands.

TABLE B-4 SunScreen EFS 3.0 Configuration Editor `ssadm edit` Sub-Command Summary

`edit` **Sub-Command**	**Description**
`add`	Create or redefine an entry.
`add_member`	Add member to an Address, Certificate, or Service group.
`authuser`	Manipulates the list of authorized users.
`del[ete]`	Delete the specified entry of the given TYPE.
`del[ete]_member`	Delete member from an Address, Certificate, or Service group.
`insert`	Insert a new object of one of the ordered (indexed) types in a specified position in the corresponding list.
`jar_hash`	Manipulates the list of JAR hashes used by the HTTP proxy.
`jar_sig`	Manipulates the list of JAR signatures used by the HTTP proxy.
`list`	Display all data for all entries or a specific entry of a give TYPE.
`list_name`	Display the set of unique basenames and sub-type of all of a given TYPE.
`load`	Load a policy into the configuration editor.
`lock`	Lock the Registry and policy in anticipation of performing edits.
`lock_status`	Return the status of the lock relative to this editor.
`mail_relay`	Manipulates the list of mail relays used by the SMTP proxy.
`mail_spam`	Manipulates the list of spam domains used by the SMTP proxy.
`move`	Move an indexed entry from its current location in the ordered list to the new location.
`proxyuser`	Manipulates the list of proxy users.
`refer`	Determine if a named-object of a given TYPE is referred to in the Registry or the current policy.
`referlist`	Display a list of all entries in the Registry or the current policy that refer to a specified named-object of a given TYPE.

TABLE B-4 SunScreen EFS 3.0 Configuration Editor `ssadm edit` Sub-Command Summary

`edit` **Sub-Command**	**Description**
`reload`	Discard any and all edits, if made, and reload the data into the editor from the database.
`rename`	Rename a specified named-object of a given TYPE.
`renamereference`	Renames all references to a specified named-object of a given TYPE.
`replace`	Replace an object at a specified index.
`save`	Save all current edits to the Registry and policy.
`search`	Search the Registry for objects that match specified criteria.
`vars`	Manipulates variables used for RADIUS configuration.
`verify`	Takes no arguments and verifies the currently loaded policy.
`quit`	Cause the editor to terminate if there are no unsaved changes.
`QUIT`	Cause the editor to terminate even if there are unsaved changes.

In the following command descriptions, when the name of an object of a particular TYPE is required, it is indicated by *name_TYPE*. If an index is needed, it is indicated by #. A keyword that was required in previous SunScreen releases but is now optional, is indicated by *KEYWORD*.

`add` Sub-Command

Creates or re-defines 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 Sub-Command

add address "*name_ADDRESS*" <HOST> #.#.#.#

add address "*name_ADDRESS*" <RANGE> #.#.#.# #.#.#.#

add address "*name_ADDRESS*" <GROUP> { "*name_ADDRESS*" ... } { "*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 Sub-Command

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)

CDP {"on" if present, "off" otherwise}

ROUTING {"on" if present, "off" otherwise}

DNS {"on" if present, "off" otherwise}

NIS {"on" if present, "off" otherwise}

LOGSIZE # {default is 100MB if not present}

SPF #.#.#.# #.#.#.# {Network and Netmask for stealth type Interfaces}

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 part of an HA cluster, and administered remotely, then the following fields must be specified as well. They can be specified in any order *after* the "screen" keyword:

ADMIN_IP #.#.#.#

ADMIN_CERTIFICATE "*name_CERTIFICATE*"

KEY "*name_KEY_ALGORITHM*"

DATA "*name_DATA_ALGORITHM*"

MAC "*name_MAC_ALGORITHM*"

COMPRESSION "*name_COMPRESSION_ALGORITHM*"

TUNNEL "*name_ADDRESS*"

The screen "*" is reserved and cannot be edited.

add service Sub-Command

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, and each Discriminator list must also 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 #-# (Note, no space is allowed before or after the "-" character)

BROADCAST #

BROADCAST #-#

An optional parameter for discriminators, which appears immediately *after* the 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*"

add interface Sub-Command

add interface "*name_INTERFACE*" type "*name_ADDRESS*"

type must be one of ADMIN, DISABLED, EFS, HA, or SPF.

The following fields are optional for stealth interface types and can be specified in any order *after* the "interface" keyword. Up to 5 ROUTERs per stealth interface can be specified. More can be specified, but only 5 (no guarantee which ones) are used by the system.

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 DETAILED

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 Sub-Command

add certificate "*name_CERTIFICATE*" SINGLE NSID # MKID "#"
add certificate "*name_CERTIFICATE*" GROUP "*name_CERTIFICATE*" ...

For GROUP certificates, a space-separated list of "*name_CERTIFICATE*" entries follows the GROUP keyword.

The following fields are optional for SINGLE entries and may be specified in any order *after* the "certificate" keyword:

G #

P #

KEY_LENGTH #

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 time Sub-Command

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*"

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 }". 24-hour time format is used, so the valid times are 0:00 (starting at midnight) through 24:00 (ending at midnight).

add rule Sub-Command

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 DETAILED {also LOG_DETAILED}

LOG SESSION {also LOG_SESSION, only valid for ALLOW rules, will be error for DENY}

SNMP {"on" if present, "off" otherwise}

USER "*name_USER*" {only used if PROXY_FTP or PROXY_Telnet set below }

TIME "*name_TIME*"

SCREEN "*name_SCREEN*"

COMMENT "*comment string*"

The following combo-field 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*"

The following fields are optional and only valid within a SKIP_VERSION_1 or SKIP_VERSION_2 combo-field. They can be specified in any order *after* the SKIP_VERSION_# keyword:

SOURCE_TUNNEL "*name_ADDRESS*"

DESTINATION_TUNNEL "*name_ADDRESS*"

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 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 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}

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}

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 Sub-Command

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 Sub-Command

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 Sub-Command

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*"

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 Sub-Command

add vpngateway *"name_VPN"* *"name_ADDRESS"* SKIP *"name_CERTIFICATE"*

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"*

The following fields are optional and can be specified in any order *after* the "vpngateway" keyword:

TUNNEL *"name_ADDRESS"*

COMMENT *"comment string"*

add_member Sub-Command

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"*
```

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"
```

authuser Sub-Command

Manipulates the list of authorized users.

Usage:

```
authuser add name parameters...
authuser delete name
authuser print
```

delete Sub-Command

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 #
```

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 Sub-Command

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*”*

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 Sub-Command

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 Sub-Command

Manipulates the list of JAR hashes used by the HTTP proxy.

Usage:

`jar_hash add` *name hash*

`jar_hash delete` *name*

`jar_hash rename` *oldname newname*

`jar_hash list`

`jar_hash list_names`

Functions:

`add` — Add an entry to the `jar_hash` database.

`del` — Delete an entry from the `jar_hash` database.

`rename` — Rename an entry in the `jar_hash` database.

`list` — List the entries in the `jar_hash` database.

`list_names` — List the names of the entries in the `jar_hash` database.

jar_sig Sub-Command

Manipulates the list of JAR signatures used by the HTTP proxy.

Usage:

`jar_sig add` *name sig-hash*

`jar_sig delete` *name*

`jar_sig rename` *oldname newname*

`jar_sig list`

`jar_sig list_names`

Functions:

`add` — Add an entry to the `jar_sig` database.

`del` — Delete an entry from the `jar_sig` database.

`rename` — Rename an entry in the `jar_sig` database.

`list` — List the entries in the `jar_sig` database.

`list_names` — List the names of the entries in the `jar_sig` database.

`list` Sub-Command

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 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 following field is optional and can be specified after the *TYPE* keyword in any of the above requests except those which are preceded by an "*".

```
SCREEN "name_SCREEN"
```

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 Sub-Command

Displays the set of unique basenames and sub-type 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, stateengine, interface, certificate, time, or vpn.

load Sub-Command

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 Sub-Command

`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
```

`lock_status` Sub-Command

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, in which case, 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` Sub-Command

Searches the Registry for objects that match specified criteria.

Usage:

`search` *`TYPE`* `[SCREEN "`*`name_SCREEN`*`"] [ SUBTYPE` *`subtype`*`]` *`Substring...`*

TYPE can be any of address, screen, service, stateengine, interface, certificate, or time. SUBTYPE values depend upon the *TYPE* being searched according to the following table.

TABLE B-5 Search *TYPE*

TYPE	SUBTYPE
address	HOST, RANGE, GROUP
certificate	SINGLE, GROUP
screen	
stateengine	
service	SINGLE, GROUP
interface	ADMIN, DISABLED, EFS, HA, SPF
time	

move Sub-Command

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 Sub-Command

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. A short-hand 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 Sub-Command

Determines if a named-objects of a given TYPE is referred to in the common data or the current policy.

Usage:

```
refer address "name_ADDRESS"
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` Sub-Command

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 screen` *"name_SCREEN"*

`referlist service` *"name_SERVICE"*

`referlist stateengine` *"name_STATEENGINE"*

`referlist certificate` *"name_CERTIFICATE"*

`referlist time` *"name_TIME"*

`referlist vpn` *"name_VPN"*

`rename` Sub-Command

Renames a specified named-object of a given TYPE.

Usage:

`rename address` *"name_ADDRESS" "name_ADDRESS"*

`*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 Sub-Command

Renames all references to a specified named-object of a given TYPE.

Usage:

```
renamereference address "name_ADDRESS" "name_ADDRESS"
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 Sub-Command

Saves all current edits to the common objects and policy.

Usage:

```
save
save "name_POLICY" [types...]
```

If a name is specified, then the current policy is written to the new name, but remains the policy in the editor.

The following types can be specified.

- Rule
- Nat
- AccessLocal
- AccessRemote
- VPNGateway

reload Sub-Command

Discards any and all edits (if any were made) and re-loads 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, and the current editor process wants to perform edits, it must perform a reload first, to re-acquire its read lock and ensure that no saved edits are lost.

Usage:

```
reload
```

verify Sub-Command

Takes no arguments and “verifies” the currently loaded policy.

Usage:

```
verify
```

mail_relay Sub-Command

Manipulates the list of mail relays used by the SMTP proxy.

Usage:

```
mail_relay parameters
```

mail_spam Sub-Command

Manipulates the list of spam domains used by the SMTP proxy.

Usage:

```
mail_spam add domain-string
mail_spam del domain-string
mail_spam list
```

proxyuser Sub-Command

Manipulates the list of proxy users.

Usage:

```
proxyuser add name parameters...
proxyuser delete name
proxyuser print
```

vars Sub-Command

Manipulates variables used for RADIUS configuration.

Usage:

```
vars add variables
```

quit Sub-Command

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 Sub-Command

`QUIT` (typed in upper case) causes the editor to terminate even if there are unsaved changes.

Usage:

```
QUIT
```

Network Monitoring and Maintenance

Examining Logged Packets (*ssadm logdump*)

SunScreen EFS 3.0 provides flexible logging of packets. A packet can be logged when it matches a rule, or SunScreen EFS 3.0 can be configured to log packets that do not match any particular rule. Most frequently, packets matching Fail rules or packets that are dropped because they do not match any rule are logged. The *action* definition of a rule controls whether a packet is logged and whether the logging is detailed or summary.

To set the logging type of packets being dropped because they do not match any rule, use the administration GUI.

Examining logged packets can be a very useful tool in troubleshooting problems in setting up configurations. For example, when first creating configurations, make the default Fail action “log packets.” This way, the logs can be reviewed to discover forgotten protocols that then can be added to the configuration. A system administrator can also use logging to capture any attempts to break in.

Logs are retrieved and cleared using the Logs page of the administration GUI. Once a log is retrieved, it can be examined using the `ssadm logdump` command.

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`, `ssadm logdump` also adds additional information such as the interface on which the packet was received and the reason that the packet was logged.

For details on the `ssadm logdump` command, refer to the `ssadm logdump` man page, by typing:

```
man logdump < log_file
```

to run `ssadm logdump` and display packets in the log. *log_file* is a log file that is downloaded from the Screen.

Session Records

When the `ssadm logdump` program is run with the `loglvl sess` filter expression, it outputs the data about sessions in the log files.

Sessions can be either TCP sessions (connections), UDP sessions (request/response pairs), or IP sessions (traffic of a particular IP type between a pair of hosts).

These session data track the state saved in the state tables in the firewall. Session data are enabled by setting the log type to "session" for encryption and pass rules.

The format of the log entries is as follows:

`TCP session: ID` *id* `SRC` *srcaddr*`:`*srcport* `DST` *dstaddr*`:`*dstport* `FWD` *fwdpackets*`:`*fwdbytes* `REV` *revpackets*`:`*revbytes* `TIME` *starttime*`:`*stoptime* `STATE` *finalstate*

`UDP session: ID` *id* `SRC` *srcaddr*`:`*srcport* `DST` *dstaddr*`:`*dstport* `FWD` *fwdpackets*`:`*fwdbytes* `REV` *revpackets*`:`*revbytes* `TIME` *starttime*`:`*stoptime*

`IP session: ID` *id* `SRC` *srcaddr* `DST` *dstaddr* `PROTO` *proto* `FWD` *fwdpackets*`:`*fwdbytes* `REV` *revpackets*`:`*revbytes* `TIME` *starttime*`:`*stoptime*

where:

TABLE B-6 Session Record Arguments

Argument	Description
id	ID for the session. If two sessions have the same ID, then they are somehow associated with each other. For example, an FTP control and data session, or a RealAudio™ control and audio session.
srcaddr	IP source address of the session either as a name or address.
srcport	Source port of the session.
dstaddr	IP destination address of the session either as a name or address.
dstport	Destination port of the session.
proto	IP protocol for IP sessions; for example, 6 = TCP.
fwdpackets	Number of packets sent in the forward direction. That is, packets from the source address to the destination address.
fwdbytes	Number of bytes sent in the forward direction. That is, packets from the source address to the destination address.
revpackets	Number of packets sent in the reverse direction. That is, packets from the destination address to the source address.
revbytes	Number of bytes sent in the reserve direction. That is, packets from the destination address to the source address.

TABLE B-6 Session Record Arguments

Argument	Description
starttime	Start time of the session in seconds since midnight GMT Jan 1, 1998.
stoptime	Stop time for the session in seconds since midnight GMT, Jan 1, 1998. You can calculate total session elapsed time by subtracting *starttime* from *stoptime*.
finalstate	Binary value representing the final state of TCP connections. The following values are possible: 1. A connection was not established because no response to SYN packet was received from the server. 2. A connection was not established because no response to the SYN / ACK packet was received from the client. A large number of these sessions could indicate a SYN attack. 3. A connection timed out due to lack of traffic. 4. A connection closed successfully or was reset.

Maintaining Your SunScreen Network

Installing a Patch

If and when patches are necessary, follow the instructions accompanying the patch.

Troubleshooting

If you have access to the console on your SunScreen EFS 3.0 (through a serial line or directly connected CRT), you can use the `ssadm debug_level` command to control the printing of debugging information from the SunScreen EFS 3.0 kernel.

Using the `ssadm debug_level` Command

If you type `ssadm debug_level` with no arguments, it 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`, it sets the kernel debugging mask to that level. To get a list of debugging bit choices type

```
ssadm debug_level -h
```

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 EFS 3.0 because they do not match any rule are reported. This is a quick way to see if the SunScreen EFS 3.0 is passing packets that you expect it to pass. You can also achieve this same result by setting the default action for 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.

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/contacting`.

It is helpful to first gather information describing your configuration. This information can be collected by saving the output of the following SunScreen EFS 3.0 `support` commands. You invoke these commands for infomation that is useful in troubleshooting through the `ssadm lib/support` command.

The `support` command has the form:
`ssadm [ -r` *Name_of_the_Screen* `] lib/support` *function parameters ...*

See the Unstable Command section earlier in this appendix.

More Details About Creating New Services

In most cases, the predefined SunScreen EFS 3.0 services provide all of the service definitions needed to create the rules. However, your particular version may need to define a new *Single Service* for your environment. A new single service can be easily defined using the list of protocol engines provided.

A brief description of other services you may need to add or to modify for your environment follows:

- IP Packets
- ICMP Packets
- TCP Services
- UDP Protocols
- NTP Traffic
- Archie Traffic
- RCP Traffic

IP Packets

SunScreen EFS 3.0 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 above state engines, especially with protocol specified as "*" (any protocol), is very dangerous. They should only be used in special cases or if the data are part of an encrypted tunnel.

Note – The predefined IP services do not pass broadcast traffic. If you wish to pass broadcast traffic, you must define a new service or add broadcast to the predefined service.

ICMP Packets

SunScreen EFS 3.0 provides predefined services for screening ICMP packets including `ping`.

These services are built upon the `icmp` state engine and allow ICMP `ping` request-and-response exchange to occur between a Source and Destination system. Use the predefined service `ping` if you want to provide `ping` access.

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.

- **Example**:

Service	Source	Destination	Action
`ping`	Inside	Outside	accept
`icmp-unreach`	Outside	Inside	accept

The above rules allow Inside machines to ping Outside machines, but not vice versa. It also allows ICMP unreachable packets to be sent from Outside machines to Inside machines. 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).

TCP Services

SunScreen EFS 3.0 screens TCP services by TCP destination port. Most common TCP services have been predefined in the services entries supplied with SunScreen EFS 3.0.

If you need to define a new TCP service, define a new service entry specifying the `tcp` filter state machine. Specify the well-known destination *TCP port(s)* of the service you wish to pass. If you specify "*" for the port, this means to pass *all* TCP services regardless of port. Note that some services such as FTP and RSH cannot be passed in this way since they are not simple TCP protocols because they make additional connections made in the reverse direction. They must be specified as separate services if you wish to pass them.

The TCP state engine times out unused and silent connections. Currently, this *time-out* is set to five hours after a connection has been established. Since some systems repeatedly retransmit until receiving some sort of error on terminated TCP connections, you might wish to enable sending ICMP rejects on illegal TCP connections, especially on your internal interfaces.

- **Example:**

Service	*Source*	*Destination*	*Action*
`telnet`	Inside	Outside	normal

The above rule allows `telnet` connections to be made from Inside machines to Outside machines.

UDP Protocols

SunScreen EFS 3.0 contains several state engines to handle UDP protocols. They are:

- `udp`—Stateful UDP packet filtering. Allows a single request-and-response exchange between Source and Destination. State entries are timed out in 20 seconds if no response is received.
- `udpall`—This state engine is identical to `udp`, but has a lower precedence. (For an explanation of precedence, see Appendix C of this *SunScreen EFS 3.0 Reference Manual*.) It is useful for avoiding conflicts while defining service groups containing many services.
- `udp_datagram`—This engine passes UDP packets in *one* direction; from Source to Destination. You can specify that broadcast packets should be passed.
- `udp_stateless`—This state engine allows UDP packets to be sent between Source and Destination. The field *UDP Port(s)* 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. Because some services use unreserved port numbers, use of this state engine can open up security holes. We do not recommend its use.

With all of the UDP engines, you define a new service entry specifying the well-known destination, *UDP port*. Specifying port "*" passes *all* UDP traffic.

NTP Traffic

SunScreen EFS 3.0 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 predefined service `ntp`.

Broadcast NTP is not supported.

Archie Traffic

SunScreen EFS 3.0 contains a service definition to handle the Archie UDP protocol. To screen Archie traffic, use the predefined service `archie`.

RPC Traffic

SunScreen EFS 3.0 contains a state engine to handle the RPC protocols. This can safely screen RPC protocols as long as they use the portmapper and do not use dynamic RPC program values.

If you need to define a new RPC service, define 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*, this means it 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. Since a lock manager makes connections in both directions (to NFS server and from NFS server) you m ay need to use the "`nlm`" service when you allow NFS access.

- **Example**

Service	Source	Destination	Action
`nfs`	Inside	DMZ	accept
`nlm`	DMZ	Inside	accept

Broadcast port mapping (NIS) is not supported for encrypted connections.

APPENDIX **C**

Services and State Engines

This appendix describes the standard services, service groups, and state engines supported by SunScreen EFS 3.0.

Standard Services

SunScreen EFS 3.0 is shipped with a number of predefined network *services*. The following table lists the services in SunScreen EFS 3.0, along with the state engine and discriminator (port, RPC program number, or type) for each service. Parameters (state engine modifiers, such as timeouts) and BROADCAST are indicated where applicable.

Service information is stored in the common object registry. See "`add service`" command in the configuration editor.

TABLE C-1 SunScreen EFS 3.0 Services

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
`echo`	`tcp`	port 7		
`discard`	`tcp`	port 9		
`systat`	`tcp`	port 11		
`daytime`	`tcp`	port 13		
`quote`	`tcp`	port 17		
`chargen`	`tcp`	port 19		
`ftp`	`ftp`	port 21		
`telnet`	`tcp`	port 23		

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
smtp	tcp	port 25		
time	tcp	port 37		
whois	tcp	port 43		
nicname	tcp	port 43		
dns	tcp	port 53		
	dns	port 53		
tftp	udp	port 69 parameters (60 –1 7)		
gopher	tcp	port 70		
finger	tcp	port 79		
www	tcp	port 80		
pop	tcp	ports 109–110		
auth	tcp	port 113		
ntp	udp	port 123		
nntp	tcp	port 119		
snmp	tcp	port 161		
	udp	port 161		
snmp traps	udp_datagram	port 162		
rlogin	tcp	port 513		
rsh	rsh	port 514		
syslog	udp_datagram	port 514		
printer	tcp	port 515		
rip	udp_datagram	port 520		
		port 520 (BROADCAST)		
sqlnet	sqlnet	port 1521		
archie	udp	port 1525 parameters (360 -1 0)		

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
certificate discovery	udp	port 1640 parameters (60 1 1)		
remote administration	tcp	ports 3852–3853		
SecurID PIN	tcp	port 3855		
HA administration	tcp	port 3856		
HA heartbeat	ping	port 8		
HA	tcp	port 3856		
securid	udp	port 5500		
securidprop	tcp	port 5510		
real audio	realaudio	port 7070		
traceroute	udp_datagram	ports 33430–34000		
			icmp	type 11
			icmp	type 3
icmp echo-reply	icmp	type 0		
icmp unreach	icmp	type 3		
icmp quench	icmp	type 4		
icmp redirect	icmp	type 5		
icmp echo-request	icmp	type 8		
router announcement	icmp	type 9		
		type 9 (BROADCAST)		
router solicitation	icmp	type 10		
		type 10 (BROADCAST)		
icmp exceeded	icmp	type 11		
icmp params	icmp	type 12		
icmp info	icmp	types 13 14 15 16 17 18		

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
ping	ping	port 8		
router discovery	icmp	type 10		
		type 10 (BROADCAST)		
			icmp	type 9
				type 9 (BROADCAST)
rstat	rpc_udp	prgm no. 100001		
	pmap_udp	prgm no. 100001		
rusers	rpc_udp	prgm no. 100002		
	pmap_udp	prgm no. 100002		
nfs prog	pmap_udp	prgm no. 100003		
	udp	port 2049		
	tcp	port 2049		
nfs readonly prog	pmap_udp	prgm no. 100003		
	nfsro	port 2049		
ypserv	nis	port 100004		
	pmap_nis	prgm no. 100004		
	pmap_nis	prgm no. 100004 (BROADCAST)		
mountd	rpc_udp	prgm no. 100005		
	pmap_udp	prgm no. 100005		
ypbind	rpc_udp	prgm no. 100007		
	pmap_udp	prgm no. 100007		
wall	rpc_udp	prgm no. 100008		
	pmap_udp	prgm no. 100008		
yppasswd	rpc_udp	prgm no. 100009		
	pmap_udp	prgm no. 100009		
rquota	rpc_udp	prgm no. 100011		
	pmap_udp	prgm no. 100011		

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
spray	rpc_udp	prgm no. 100012		
	pmap_udp	prgm no. 100012		
rex	rpc_udp	prgm no. 100017		
	pmap_udp	prgm no. 100017		
klm	rpc_udp	prgm no. 100020		
	pmap_udp	prgm no. 100020		
nlm	rpc_udp	prgm no. 100021		
	pmap_udp	prgm no. 100021		
			rpc_udp	prgm no. 100021
			pmap_udp	prgm no. 100021
status	rpc_udp	prgm no. 100024		
	pmap_udp	prgm no. 100024		
ypupdate	rpc_udp	prgm no. 100028		
	pmap_udp	prgm no. 100028		
nfs acl	rpc_udp	prgm no. 100227		
	pmap_udp	prgm no. 100227		
ospf	ip	type 89 (BROADCAST)		
skip	iptunnel	type 57		
		type 79		
icmp all	icmp	*		
		* (BROADCAST)		
ip all	ip	*		
ip mobile	ipmobile	*		
ip tunnel[3]	iptunnel	*		
ip forward	ipfwd	*		
udp all	udpall	*		
tcp all	tcpall	ports 0–3850		
		ports 3854–65535		
rpc all	rpc_udp	*		

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
rpc tcp all	rpc_tcp	*		
pmap udp all	pmap_udp	* (BROADCAST)		
pmap tcp all	pmap_tcp	*		
X11	tcp	ports 6000–6063		
pcnfsd	pmap_tcp	prgm no. 150001		
	pmap_udp	prgm no. 150001		
	rpc_tcp	prgm no. 150001		
	rpc_udp	prgm no. 150001		
automount	pmap_tcp	prgm no. 300019		
	pmap_udp	prgm no. 300019		
	rpc_tcp	prgm no. 300019		
	rpc_udp	prgm no. 300019		
ypxfrd	pmap_tcp	prgm no. 100069		
	pmap_udp	prgm no. 100069		
	rpc_tcp	prgm no. 100069		
	rpc_udp	prgm no. 100069		
exec	tcp	prgm no. 512		
wais	tcp	port 210		
uucp	tcp	port 540		
irc	tcp	port 6670		
	tcp	port 6680		
VDOLive	tcp	port 7000		
	tcp	port 7010		
			udp	port 32649
CU See Me		ports 7648–7652		
	udp_datagram			
Vosaic	tcp	port 1235		
			udp_datagram	ports 61801–61820
			udp_datagram	ports 20000–20020
StreamWorks	udp_datagram	port 1558		
			udp_datagram	port 1558

TABLE C-1 SunScreen EFS 3.0 Services *(Continued)*

Service	State Engine (forward filtering)	Discriminator	State Engine (reverse filtering)	Discriminator
CoolTalk	tcp	ports 6499–6500		
	udp_datagram	port 13000	udp_datagram	port 13000
Backweb	udp	port 370 parameters (60 0 3)		;
radius	udp	port 1645		
ssl	tcp	port 443		
who	udp_datagram	port 513 (BROADCAST)		
netstat	tcp	Port 15		
biff	udp_datagram	port 512 (BROADCAST)		
bootp	udp	port 67 (BROADCAST) parameters (60 0 3)		
kerberos	udp	port 88		
ntp-tcp	tco	port 123		
netbios name	udp	port 137		
netbios datagram	udp_datagram	port 138		
netbios session	tcp	port 139		
lpd	tcp	port 2766		
echo-udp	udp	port 7		
discard-udp	udp	port 9		
time-udp	udp	port 37		
daytime-udp	udp	port 13		
tcp-high-ports	tcp	ports 1024–65535		
udp-high-ports	udp	ports 1024–65535		

`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 EFS 3.0 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, `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.

`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.

`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. For example:

```
(old way using ip all)
"ip all" host1 host2 allow
"ip all" host2 host1 allow

(new way 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 EFS 3.0 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, since any traffic would be allowable. State engines should only be used in special cases or if the data are part of an encrypted tunnel.

Note that the predefined IP services do not pass broadcast traffic. If you want to pass broadcast traffic, you must define a new service or add broadcast to the predefined service.

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.

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
```

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.

smtp (Electronic Mail) Service

Simple Mail Transfer Protocol (SMTP) is used to send electronic mail between two message transfer agents using TCP. SunScreen EFS 3.0 includes a predefined service definition, smtp, to send and receive SMTP mail on TCP port 25.

www (World-Wide-Web Access) Service

The World Wide Web provides a graphical user interface that lets users browse a global network of services and documents. SunScreen EFS 3.0 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 will not be able to 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 machines. Instead, since you know which port your Web server uses (generally 80), you should use a more restrictive service rule, such as the `www` service definition.

`dns` Service

DNS traffic consists of both UDP and TCP traffic. SunScreen EFS 3.0 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.

`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 EFS 3.0 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.

It is usually sufficient to enable RIP in the default rule that passes RIP from the routers to all other addresses. This lets the SunScreen EFS 3.0 send and receive RIP packets without restriction. If you want to restrict RIP traffic, do not enable RIP using the default access rules; instead, define rules for RIP based on your security policy.

Service	Source	Destination	Action
`route`	*routers*	*	allow
`route`	*	*routers*	allow

sqlnet Services

SunScreen EFS 3.0 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 multi-threaded servers and pre-spawned 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 EFS 3.0 supports both types of SQL*Net connections.

realaudio Services

SunScreen EFS 3.0 contains a service definition to handle RealAudio™ sessions. To screen RealAudio traffic, use the `realaudio` service.

icmp Services

SunScreen EFS 3.0 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 following table:

Service	Source	Destination	Action
`ping`	Inside	Outside	allow
`icmp-unreach`	Outside	Inside	allow

The above rules allow Inside machines to `ping` Outside machines, but block Outside machines from sending ping messages to Inside machines. It also allows ICMP unreachable packets to be sent from Outside machines to Inside machines. 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).

TCP Services

SunScreen EFS 3.0 screens TCP services by destination port numbers. Most common TCP services are already defined in the service entries supplied with SunScreen EFS 3.0.

If you need to define a new TCP service, define a new service entry specifying the `tcp` filter state machine. 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 since they are not simple TCP protocols; they make additional connections made 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. Since some systems repeatedly retransmit until they receive an error about a terminated TCP connection, you should configure a rule using the tcp service to send an ICMP rejection message, especially on your internal interfaces.

For example, the following rule allows `telnet` connections to be made from Inside machines to Outside machines.

Service	Source	Destination	Action
`telnet`	Inside	Outside	allow

UDP Services

SunScreen EFS 3.0 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. 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.

`ntp` Service

SunScreen EFS 3.0 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.

`archie` Service

SunScreen EFS 3.0 contains a service definition to handle the Archie UDP protocol. To screen Archie traffic, use the `archie` service.

`rpc` Service

SunScreen EFS 3.0 contains a state engine to handle the RPC protocols. This can safely screen RPC protocols, 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. Since a 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.

Service	Source	Destination	Action
`nfs`	Inside	DMZ	allow
`nlm`	DMZ	Inside	allow

Broadcast port mapping (NIS) is not supported for encrypted connections.

Network Service Groups

Network services can be organized into *service groups*, so that a single rule can apply to multiple network services. TABLE C-2 lists the predefined service groups in SunScreen EFS 3.0 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.

TABLE C-2 SunScreen EFS 3.0 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`

TABLE C-2 SunScreen EFS 3.0 Network Service Groups *(Continued)*

Service Group Name	Member Services
discard	discard discard-udp
echo	echo echo-udp
HA	HA heartbeat HA administration
mosaic	www ssl gopher ftp archie
netbios	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

State Engines

SunScreen EFS 3.0 includes a number of *state engines*, which act as a protocol checker 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 this state engine. Note that since 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 one parameter:

- Response timeout in second – Specifies the time to wait for DNS responses. Default is 60 seconds.

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. This parameter specifies the lifetime of an idle control session. (Default = 600 seconds)
- Lifetime of idle data session in seconds. This parameter 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 uni-directional IP traffic of a particular IP type. The data can only flow in the forward direction (From to To 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 uni-directional IP traffic of a certain IP type. The data can only flow in the forward direction (From to To 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 (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, since 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 it is mapping to DNS entries. (Default = 2)

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. Normally, 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. Since 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 RCP 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). 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. The discriminator for the sqlnet state engine is the port number of the Oracle listener, which is normally TCP port 1521.

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.

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. Since 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 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 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, but has been replaced in most cases with stateful UDP filtering. Note that since it is stateless UDP packet filtering, services defined using this engine 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.

CHAPTER **D**

Error Messages

This appendix describes the error messages generated by various components of the SunScreen EFS 3.0 software and suggested solution for each error condition.

- Error messages from `ssadm edit`
- Error messages from `ssadm activate`
- Error messages from `ssadm lock`
- Logged packet reasons

Error Messages From `ssadm edit` Component

The `ssadm edit` component's error messages follow:

- `Usage: ssadm edit [-beim] [-I ip] [-U user] policy [-c command]`

  Return code: 1

  Indicates that the user invoked the `edit` program incorrectly.

- `Error illegal policy name` *policyname*

  Return code: 1

  Indicates that the user specified an illegal policy name while invoking the editor.

- `Discarding unsaved changes`

  Return code:

  Indicates that the user chose to QUIT even though there are unsaved changes.

- `Write Lock Held`

  Return code:

  Indicates that the user asked for their “lock_status”, and they currently hold the write lock.

- Read Lock Held

  Return code:

  Indicates that the user asked for their “lock_status”, and they currently hold the read lock.

- Registry Version: #
  Policy Version: #

  Return code: 0

  Indicates that the user asked for the “version” currently being edited.

- no longer supported. use edit del request syntax

  Return code: 251

  Indicates that the user tried to delete a NAT Rule using the old `ssadm nat` or `ssadm access` command.

- There are unsaved changes

  Return code: 231

  Indicates that the user attempted to “`quit`” without saving unsaved changes.

- Cannot name Policy “lock”

  Return code: 232

  Indicates that the user attempted to save the policy as “lock”, which is a reserved word.

- There are unsaved changes “Version.”

  Return code: 233

  Indicates that the user attempted to save the policy as “Version”, which is a reserved word.

- There are unsaved changes in “Registry”

  Return code: 234

  Indicates that the user attempted to save the policy as “Registry”, which is a reserved word.

- Cannot save a versioned file with a new name

  Return code: 235

  Indicates that the user attempted to save a versioned policy with a new name.

- Registry objects redefined

  Return code:

  Indicates that an entry in the Registry (on disk) has more than one definition. All definitions after the first are lost upon the next "`save`."

- Registry objects redefined

  Return code:

  Indicates that 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

  Indicates that the policy given to be read did not exist.

- parse error

  Return code: 241

  Indicates that the Policy or Registry file on disk is corrupt and cannot be read. Make sure you have a back-up or a recent version saved.

- could not acquire read lock

  Return code: 242

  Indicates that 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

  Indicates that a request to gain the write lock failed. Likely because some other process currently holds the write lock.

- lock not held

  Return code: 244

  Indicates that an attempt was made to "`save`" changes, but something has happened so that this process no longer holds the write lock. Perhaps someone else has issued a `ss_lock -c` *policy* and invalidated the lock.

- cannot modify "*"

  Return code: 247

  Indicates that 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

  Indicates that 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 to fix the situation.

- unresolved references

  Return code: 250

  Indicates that a reference is made to a named object in the global registry that does not exist in the registry.

- invalid input

  Return code: 251

  Indicates that a request was not well-formed.

- unknown operation

  Return code: 253

  Indicates that a request used an invalid operation.

- unknown data type

  Return code: 252

  Indicates that a request was issued for an invalid data type.

- internal error

  Return code: 255

  No longer used.

- `Error: invalid input`

  Return code: non-zero

- `Warning: Adding ADMIN Interface to an routing machine.`

  Indicates you added an ADMIN Interface to a routing machine. You probably want this to be an routing type Interface.

- `Warning: Adding ADMIN Interface to an routing machine.`

  Indicates you added an ADMIN Interface to an routing machine. You probably want this to be an routing type Interface.

- `(ssadm interfaces) Warning: operation replaced the Administrative Interface`

  Only appears as a result of an “add Interface” request

- `(ssadm interfaces) Warning: operation replaced only EFS Interface`

  Only appears as a result of an “add Interface” request

- `(ssadm interfaces) Warning: operation replaced only SPF Interface`

  Only appears as a result of an “add Interface” request

- `(ssadm interfaces) Warning: operation left only one SPF Interface`

  Only appears as a result of an “add Interface” request

- `Error: certificate invalid input`

  Return code: non-zero

Error messages can arise 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 activate` Component

The `ssadm activate` component’s error messages follow:

- `Error output directory does not exist` *output directory*

  Return code: non-zero

  Indicates that the policy being compiled and activated refers to more than 31 distinct Time objects.

- Too many Time objects being used. Limit is 31.

  Return code: 236

  Indicates that the user invoked `ssadm activate` incorrectly.

- Registry objects redefined

  Return code:

  Indicates that 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

  Indicates that the `-S` passed to `ssadm activate` is a non-existent Screen object.

- file not found

  Return code: 240

  Indicates that the policy given to be read did not exist.

- Too many Time objects being used. Limit is 31.

  Return code: 236

  Indicates that the user invoked `ssadm activate` incorrectly.

- parse error

  Return code: 241

  Indicates that the Policy or Registry file on disk is corrupt and cannot be read. Be sure you have a back-up or a recent version saved.

- compile error

  Return code: 245

  No longer used.

- unresolved references

  Return code: 250

  Indicates that a reference is made to a named object in the global registry, which does not exist in the registry.

- Error: Status NAT, but Addresses are different sizes (NAT entry)

  Return code: non-zero

- Error: Original and Translated Source Intersect

  Return code: non-zero

- Error: Original and Translated Destination Addresses Intersect

  Return code: non-zero

- Error: Cannot translate both source and destination addresses

  Return code: non-zero

- Screen object must define smtp address

  Return code: non-zero

  Indicates that the Screen object must define the SMTP Address if the SMPT Proxy is to be used.

- Error: Service not defined (remote administration)

  Return code: non-zero

  Indicates that 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: non-zero

  Indicates that 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: non-zero

  Indicates that 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: non-zero

  Indicates that only 16 of a given type of Interface is supported.

- Could not find HA Service

  Return code: non-zero

  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: non-zero

  Indicates that the Screen objects participating in the current HA cluster lack HA_IP addresses.

- Error: HA IP address is not on HA interface

  Return code: non-zero

  Indicates that the HA IP address specified is not part of the HA Interface.

- Error: Service incorrectly defined

  Return code: non-zero

  Indicates that a service has contradictory information, such as the same port, but different state engines, or different parameters.

- Error: Interfaces intersect

  Return code: non-zero

  Indicates that two (or more) Interfaces’ addresses intersect.

- Error: Rule uses Certificate Group with Service containing Reverse Filter *RULE*

  Return code: non-zero

  Indicates that the problem is that 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: non-zero

  Indicates that the problem is that the reverse Rule swaps the Certificates, and Groups are not supported in the encrypting case.

- Error: Service not defined (remote administration)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (snmp traps)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (skip)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (certificate discovery)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (rip)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (dns)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: Service not defined (nis)

  Return code: non-zero

  Indicates that the service is needed internally, but has been either renamed or deleted.

- Error: could not generate Rule from Screen: *Screen name*

  Return code: non-zero

- Error: could not generate Rule to Screen: *Screen name*

  Return code: non-zero

- Error: *Screen name* is missing encryption parameters

  Return code: non-zero

- Error: *Screen name* requires a certificate to administer Screen *Screen name*

  Return code: non-zero

- Error: *Screen name* requires a certificate to be administered by Screen *Screen name*

  Return code: non-zero

- Error: HA Secondary with no Master

  Return code: non-zero

  Indicates that HA is indicated, but no primary Screen is specified.

- Error: Incomplete Screen definition

  Return code: non-zero

  Indicates that 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: non-zero

- Error: HA enabled with multiple HA Interface(s) defined

  Return code: non-zero

- Error: HA not enabled but HA Interface(s) defined

  Return code: non-zero

- `datacompiler: Error writing data file (fseek failed)`

  Indicates that the datacompiler could not write the output file due to a failed `fseek`.

- Error: Remote Certificate *name1* uses multiple local certificates: *name2* and *name3*

  Indicates that 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 EFS supports only using a one local certificate for any given remote certificate in SKIP_VERSION_1 compatibility mode. The user must either use skip_version_2 or change one of *name2* and *name3* to the other.

- datacompiler: Manual Table too large.
  Limit is 65535.

  Indicates that there are more than 65535 pairs of certificates (unlikely) for either Manual Keying or support SKIP_V1 nodes. There is a limit of 65535.

- datacompiler: Skip_V1 Table too large.
  Limit is 65535

  Indicates that there are more than 65535 pairs of certificates (unlikely) 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:` *NAME*
  `Activated by` *user* `on` *date*

  Return code: 0

- `Warning: Rule type could not be determined and is being discarded` *Svc addr* `. . .`

  Indicates that a problem determining how to implement the rule occurred and is being discarded.

- *TYPE*: *name* is already defined. Redefined on line <#> in file *file*

  Indicates that 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. The user needs to define *name1* or remove the reference by modifying *name2*.

- Address *name* is part of a cycle.

  Indicates 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. The user must break the cycle for the compilation to be successful.

- Service *name* is part of a cycle.

  Indicates 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. The user must break the cycle for the compilation to be successful.

- Service incorrectly defined. *name...*

  Indicates that 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. The user must redefine this service for the compilation to be successful.

- `Error: "`*name*`" is not defined. Used on line # in file` *FILE* `by` *name*

  Indicates that the user is referring to an object (address, service) that has not yet been defined. Be sure it is defined.

- `Invalid Domain Name: "`*name*`"`

  Indicates that the user has entered a domain name that has illegal characters, such as "/." Use the default domain name "default."

- `Error: Domain "`*name*`" does not exist.`

  Indicates that the user has entered a non-existent domain name. Use the default domain name "default."

- `unknown operation`

  Indicates that the user has requested an operation that is not recognized.

- `Sorry, `*`character`*` character not supported.`

  Indicates that the user has entered an unsupported character.

- `could not acquire lock to read data, please re-issue request.`

  Indicates that too many concurrent processes are running.

- `could not acquire lock to write data, please re-issue request.`

  Indicates that too many concurrent processes are running.

- `invalid input`

  Indicates that the user entered something incorrectly. Refer to the relevant man page to verify you have the correct command syntax.

- `unknown data type.`

  Indicates that the user requested an operation on an unknown data type.

- `Error: “`*`name`*`” cannot be a Local Certificate.`

  Indicates that 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.

- `Error: Missing Remote Certificate: “`*`name`*`”`

  Indicates that 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.`

  Indicates that 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`*`”`

  Indicates that 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 "*"`

  Indicates that the user 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`

  Indicates that the user indicated service is missing.

- `Warning: `*`RULE`*` uses invalid pair of certificate and will be ignored`

  Indicates that a SKIP-based rule must include one local and one non-local certificate. If both are local, or both are non-local, 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 non-local, and re-activate.

- “smtp-server” must refer to single IP Address

  Indicates that the smtp-server definition refers to an address that must contain a single IP address as its value. Verify that the specified address is either a host address or a list that contains only a host address.

- Warning: *PROXY* proxy server not found. No rule generated

  Indicates that the user 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*

  Indicates that the configuration does not exist. Try again with a configuration that does exist.

- Error: # NAT entries are incorrectly defined

  Indicates that 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

  Indicates that the service “HA Service” could not be found. Be sure it is defined.

- Could not find HA_HOSTS address

  Indicates that the HA_HOSTS address could not be found. The HA_HOSTS address contains the IP addresses of all the HA Screens in the cluster so rules can be made that allow them to share data. Be sure this address is defined.

- Service incorrectly Defined: *SERVICE*

  Indicates that 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.
  Re-define the service correctly.

- Error: Interface does not exist (le0)

  Indicates that le0 is the name of the non-existant interface. This happens if the global common registry being activated contains an Interface that the machine the compile and activate is happening on does not have .

- Warning: Could not verify Interface “*name*” exists

  Indicates that the user added an Interface and it could not be verified on the Screen.

- Warning: Adding stealth Interface to an routing Machine

  Indicates that the user added an Interface of type stealth on an routing machine. This configuration is not supported.

- Warning: Adding ADMIN Interface to an routing machine

  Indicates that the user added an ADMIN Interface to an routing machine. You probably want this to be an routing type Interface.

- `Expecting “{“ but found a` *character*

  Indicates that the syntax entered was incorrect. See the man page for correct syntax.

- `Expecting “}” but found a` *character*

  Indicates that the syntax entered was incorrect. See the man page for correct syntax.

- `Unexpected end of input`

  Indicates that the syntax entered was incorrect. See the man page for correct syntax.

- `Invalid Range specified` *# - #*

  Indicates that the user entered a range where the end value was less than the start value.

- `Service definition is internally inconsistent`

  Indicates that the user specified service is not well-defined. For example, it may specify the same port for conflicting state engines, such as `UDP` and `UDP-datagram`. Redefine the service correctly.

Error Messages From `ssadm lock` Component

The `ssadm lock` component’s error messages follow:

- Usage: ssadm lock -w | -c policy

  Return code: 1

  Indicates that the user invoked `ssadm lock command` incorrectly.

- `lock held by` *user @ IP* `process id` *pid*

  Indicates that the user used pid.

- `Lock available`

  0

Logged Packet Reasons

The following table lists common reasons for logging packets in the SunScreen EFS log and in the SNMP syslog files. Packets logged for reasons with numbers below 256 indicates that the packet passed. Packets logged for reasons with number of 256 or greater indicates that the packet was dropped.

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.
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 invalid port number specification. An example is an FTP data session not on port 20.
261	`Bad format`	`invalidFormat(261)`	Packet dropped due to invalid format. The Screen determined that the packet did not match the service specified in the rules.

Number	Log Error Message	SNMP Error Message	Explanation
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.
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).

Number	Log Error Message	SNMP Error Message	Explanation
269	`Bad interface`	`invalidInterface(269)`	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: • The packet was received encrypted, but the rule specified that it must be unencyrpted. • 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. 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.
270	`Bad policy`	`invalidPolicy(270)`	Indicates that 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.

Number	Log Error Message	SNMP Error Message	Explanation
274	`Fragment too big`	`fragmentTooBig(274)`	Indicates a possible network attack.
275	`Fragment overlap`	`fragmentOverlap(275)`	Indicates that a packet was fragmented while it was in transit and that the fragments contain redundant data. May indicate a network attack.

APPENDIX **E**

Glossary

ACL access control list. Limits and controls who uses a host system or applications through communications link.

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 EFS uses IP addresses.

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.

AnswerBook™ online documentation The Sun™ online documentation for use with the OpenWindows™ environment. See also online documentation.

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.

applet A program written in the Java™ programming language to run within the HotJava™ browser, the World Wide Web (WWW) browser.

argument Item of information following a *command*. It may, for example, modify the command or identify a file to be affected.

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.

block Groups of consecutive bits (or bytes). In the Java™ programming language, any code between matching braces ({ and }).

block cipher or block algorithm Encryption algorithm that encrypts blocks. See *stream ciphers.*

Bourne shell The *shell* used by the standard Bell Labs UNIX® developed by Steve Bourne in 1979.

broadcast *Packet* delivery system, where a copy of a given packet is distributed to all hosts attached to the network.

button One-choice element of a control area or a menu that starts an activity. Buttons execute commands (*command buttons*), display pop-up windows (*window buttons*) or a dialog box, and display menus (*menu buttons*).

CA See *Certification Authority.*

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.

CD-ROM compact disk, read only memory. Storage medium that uses laser optics rather than magnetic capability to read data.

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 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.

Certification 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.

CFB Cipher Feedback. Uses a block cipher (such as DES) to implement a stream cipher.

cipher Cryptographic algorithm used for encryption or decryption.

ciphertext Encrypted message.

cluster Screens in a HA cluster connected by a high-speed network that work together as if they were one Screen. See *high availability*.

command Instruction to the computer. A command typically is a character string typed at a keyboard and is interpreted by the computer as a demand for a particular action. In a graphical user interface (GUI), a *button*, menu item, or *control*.

command button Button used to execute application commands.

common objects Data objects that are relevant to all SunScreen EFS 3.0 policies. They include: address, screen, state engine, service, interface, certificate, time, policy, and virtual VPN gateway groups.

compiler Translation program that converts a high-level computer language into machine language.

concatenate Join together sequentially. The UNIX® cat command, for example, concatenates files.

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.

control Object in a *menu* that is used to perform an action.

cookie General mechanism that server side connections can use to store and retrieve information on the client side of the connection. That is, cookies are small data files written to the hard drive by some Web sites when viewed in a Web browser. These data files contain information the site can use to track such things as passwords, lists of pages visited, and the date when a certain page was last looked at. The term cookie originated from "fortune cookie" because it sends a different message, or "fortune," each time used.

cryptanalysis Art and science of breaking cryptographic algorithms and protocols.

cryptographic Algorithm used to keep data secure.

C shell Standard shell provided with Berkeley standard versions of UNIX®.

daemon UNIX® process that runs in the background to perform a task on behalf of the system.

data compression Application of an algorithm to reduce the space required to store or the bandwidth required to transmit data.

datagram In a packet-switching network, a message and associated Internet source and destination addresses.

decoder Facility that takes data that has been encoded, or compressed, by an *encoder* and decodes or decompresses it.

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 commonly used algorithm developed by IBM for the U.S. National Bureau of Standards for encrypting and decrypting data. See *CBC*.

Diffie-Hellman See *DH*.

DH Diffie-Hellman. Named after its inventors, DH is a classic cryptographic construction that uses exponentiations over a prime field.

digital signatures Sixteen-byte MD5 hash of an electronic document that allows the recipient to verify the integrity of the document and the identity of the sender.

diskette 3.5–inch removable storage medium supported by some Sun systems.

DMZ See *demilitarized zone*.

DN distinguished name. Numeric string representation of a list of IP addresses or equivalent identifier for principals in the network, such as IP nodes or users.

DNS domain naming system. Distributed name and address mechanism used in the Internet.

DSA Digital Signature Algorithm. Each DSA is responsible for the directory information for a single organization or organizational unit.

DST Destination addresses.

dynamic packet screening Process of examining traffic to be either allowed or denied.

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.

EFS Encrypting Firewall System.

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 *tunnel mode*.

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*.

entity In International Organization for Standardization's open systems interconnection (OSI), a layer protocol machine. An entity within a layer accesses the layer entity below and provides services locally to the layer entity above.

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.

export controlled Version of the SunScreen SKIP certificate software that uses 1024-bit keys and allows users to select DES, RC2, or RC4 for traffic encryption. Compare global and U.S. and Canada use only.

failover Process by which a passive Screen in a high availability group takes over for the active Screen if the active Screen fails or becomes unavailable.

FDDI Fiber Distributed Data Interface. High-speed networking standard. The underlying medium is fiber optics, and the topology is a dual-attached, counter-rotating *token ring*. FDDI networks can often be identified by the orange fiber "cable."

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 Command used to copy files.

FTP File Transfer Protocol. An interactive file transfer protocol often used on TCP/IP networks to copy files to and from remote computers. Requires users to log in to the remote computer.

FTP proxy Can be configured to allow or deny specific FTP commands such as PUT or GET.

gateway indication of systems that translate from one native format to another. Transfers and converts information to a receiving network. See *VPN*.

gif Graphics Interchange Format. A format for compressing bitmap files to define how the computer accesses and draws files. See *jpeg* and *tiff*.

global Version of the SunScreen SKIP certificate software that uses 512-bit keys and allows users to select RC2 or RC4 for traffic encryption. Compare export controlled and U.S. and Canada use only.

graphical user interface Provides the user with a method of interacting with the computer and its special applications, usually with a mouse or other selection device.

GUI See *graphical user interface*.

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*.

hash Message digest or cryptographic checksum.

header file File of information, identified at the beginning of the program, that contains the definitions of data types and variables used by the functions in the program.

heartbeat A periodic message sent between the two membership monitors t each other. Lack of a heartbeat after a specified interval and number of retries can trigger a takeover. 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 EFS 3.0, host is only used when referring to a *source* or *destination* of a *packet* of the network traffic being discussed.

HTML Hypertext Markup Language. A file format, based on SGML, for hypertext documents on the Internet.

HTTP Hypertext Transfer Protocol. Internet protocol that fetches hypertext objects from remote hosts. See *URL*.

HTTP proxy Can be configured to allow or deny Java applets, and Active-X 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*.

icon On-screen graphic symbol that simplifies access to a program, command, or data file. Displaying objects as icons conserves screen real estate while keeping the window available for easy access.

IDEA International Data Encryption Algorithm. Used to make a message impossible for others to read.

IEEE Institute of Electrical and Electronics Engineers. The group that produced the standards for Ethernet and Token Ring.

Interfaces Describes the physical interface ports of Screen objects.

Initial configuration When installing SunScreen EFS, the user creates, compiles, and activates a configuration named "Initial," which enables a user to connect to the SunScreen EFS 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.

International Organization for Standardization International standards-setting organization whose mandate is to foster trade between countries.

Internet Protocol Suite of protocols within TCP/IP used to link networks worldwide, developed by the United States Department of Defense and is used on the Internet. Note that this protocol suite was developed for the ARPANET, forerunner of the Internet. The prominent feature of this suite is the IP protocol. See *IP*.

IP Internet Protocol. *Network layer* protocol for the Internet Protocol suite.

ISDN Integrated Services Digital Network. Worldwide digital communications network.

ISO See *International Organization for Standardization*.

ISP Internet service provider. A company providing an Internet package. This often includes a phone number access code, username, and software—all for a provider fee.

issued certificate Certificate that is *issued* by a *Certification Authority*. See *self-generated certificate*.

ISV Independent software vendor. Third-party software developer.

Java™ Object-oriented, platform independent programming language developed by Sun Microsystems to solve a number of problems in modern programming practice. The Java language is used extensively within the HotJava™ browser.

JDK Java Development Kit. Software tools used to write Java applets or application programs.

JPEG Joint Photographic Experts Group. A format for compressing bitmap files to define how the computer accesses and draws files. See *gif* and *tiff*.

JRE Java Runtime Environment.

kernel Core system support software group used to manage the hardware and supply basic services.

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 EFS 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*.

man pages UNIX online documentation.

MD Message Digest. Authentication code that cryptographically guarantees that data has not been forged or tampered with.

MD5 Message digest one-way *hash* function designed by Ron Rivest. The algorithm produces a 128–bit hash, or message digest, of the input message.

menu List of application options.

menu button Multiple-choice control that has a *menu mark* and is used to display a menu.

menu mark Hollow triangle in the border of a button or following a menu item that has a *submenu* attached to it. The triangle points to where the menu or submenu is displayed.

MIB Management Information Base. SNMP structure that describes the particular device being monitored. See *SNMP*.

MIC Message Integrity Check.

modulus Arithmetic operation used in programming whose result is the remainder of a division operation.

multicast Special form of *broadcast* where copies of the packet are delivered to only a subset of all possible destinations.

NAT See *network address translation*.

NC See *Network Computer.*

network Hardware connecting various systems, enabling them to communicate.

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, thus allowing internal systems with unregistered addresses to access systems on the Internet.

network administrator Person who maintains a network.

network computer Connected to a network through its hardware and software.

network layer Third of the seven layers in the ISO model for standardizing computer-to-computer communications. See *ISO*.

network mask Number used by software to separate the local subnet address from the rest of a given IP address.

NFS™ Network file system. A Sun distributed file system that enables a set of computers to cooperatively access each others files in a transparent manner.

NIS and NIS+ SunOS™ 4.x network information service. NIS+ is a newer version, SunOS 5.x, with enhanced security.

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 `uname -n` 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.

NSA National Security Agency. United States of America's official cryptographic organization.

NSID Name space identifier. Used to identify a naming scheme for a key. See *key.*

OLTP On-Line Transaction Processing. Handles real-time transactions.

one-way hash Cryptographically secure hash function that cannot be reversed. See *MD5, SHA, hash.*

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.

page To advance text displayed in a window by one full screen at a time, usually using a scroll bar.

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*.

PFL Packet Filtering Language. Packet filter used by SunScreen EFS.

PFS Perfect Forward Secrecy. Captured packets that are decrypted cannot be used to decrypt other packets.

PGP Pretty Good Privacy. Public-domain email encryption program that uses *IDEA* for data encryption, RSA for key management, and MD5 as a one-way hash function. See *RSA*, and *MD5*.

PID process identification number. Unique, system-wide, identification number assigned to a process.

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. Used for TCP/IP connectivity, usually for PCs over a telephone line.

policy Named set of policy data. For example, when the SunScreen EFS 3.0 software is first installed, there is one policy, named "Initial."

policy objects Rules that define a security policy in terms of the common data objects for SunScreen EFS 3.0. Policy data include filtering rules, NAT rules, and administration access rules.

policy rules Rules that pertain to a centralized managed group or an HA cluster. See *Screen-specific rules*.

pop-up window Window that displays to perform a specific function and then is dismissed.

POSIX Portable Operating System Interface for Computer Environments. A set of standards that define the applications interface to basic system services for input/output, file system access, and process management, using the C programming language, which establishes standard semantics and syntax.

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*.

pseudo-random Pseudo-random 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 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.

query Process for extracting particular data.

quit To stop in an orderly manner; to execute the normal shutdown of a program and return control to the operating system.

RC2 and RC4 Variable-key-size encryption algorithms designed by Ron Rivest. RC2 is a variable-key-size block *cipher*, designed to be a replacement for *DES*. RC4 is a variable-key-size stream cipher that is stated to be ten times faster than DES. Both algorithms are quite compact, and their speed is independent of the key's size. See *DES*.

RC2-40 and RC4-40 Globally exportable encryption algorithms from RSA, Inc., that use 40-bit keys.

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.

RISC Reduced Instruction-Set computer. A central processing unit technology used by Sun Microsystems, Inc.

root user name SunOS user name that grants special privileges to the person who logs in with that ID. The user who can supply the correct password for the root user name is given *superuser* privileges for the particular machine.

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 sub-net 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.

RSA Popular public-key algorithm, which was named after its three inventors, Ron Rivest, Adi Shamir, and Leonard Adleman.

Screen-specific objects Data objects relevant to the policies of one Screen. See *common objects*.

SDNS Secure Data Network Service.

SDS Sun Directory Services.

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*.

security association Set of security information relating to a given network connection or set of connections.

self-generated certificate Public key value only used when entities are named using the message digest of their public value, and these names are securely communicated. See *issued certificate*.

session key Common cryptographic technique 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.

SGML Standard Generalized Markup Language. Method of tagging a document to apply to many format elements.

SHA Secure Hash Algorithm. Used to verify a digital signature.

shared-key cryptography Also known as symmetric key cryptography. Cryptography where each party must have the same key to encrypt or decrypt *ciphertext*.

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 EFS, which provides a system with the ability to encrypt any protocol within the TCP/IP suite efficiently. Once installed, systems running 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 or Pascal. The source code must be translated to machine language by a program known as the *compiler* before the computer can execute the program.

spam Electronic equivalent of junk mail.

SPARC Scalable Processor Architecture. An architecture for a family of RISC processors. See *RISC*.

special characters Also called a metacharacter, is a character having a special meaning to UNIX. For example, the UNIX shell interprets the ? character to stand for any single character.

SQL structured query language. International standard language for defining and accessing relational databases.

stack List constructed and maintained so that the next item to be retrieved and removed is the most recently stored item still in the list.

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 illegal) address. It provides a way for external hosts to initiate connections to internal hosts without actually using an external address. See *NAT*.

stealth mode Stealth-mode offers optional hardening of the OS, which removes packages and files from the Solaris operating system that are not used by SunScreen EFS 3.0. Stealth-mode requires the Screen to partition a single network.

stream algorithm or stream cipher Symmetric algorithm that operates on *plaintext* a single bit (or byte) at a time. See *block cipher*.

submenu Menu that displays additional choices displayed through a menu item on a menu.

subnet Working scheme that divides a single logical network into smaller physical networks to simplify routing.

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*.

SunCA Certification authority operated by Sun Microsystems, Inc. that issues Export-Controlled (1024-bit) certificates.

SunCAglobal Certification authority operated by Sun Microsystems, Inc. that issues Global (512-bit) certificates.

SunScreen Name of the family of security products produced by Sun Microsystems, Inc.

superuser Special user who has privileges to perform all administrative tasks on the system. Also known as *root*.

symmetric key cryptography See shared-key cryptography.

TCP See transmission control protocol.

TCP/IP Transmission Control Protocol/Internet Protocol. Protocol suite originally developed for the Internet. It is also called the *Internet* protocol suite. SunOS networks run on TCP/IP by default.

telnet Virtual terminal protocol in the *Internet* suite of protocols. Enables users of one *host* to log in to a remote host and interact as normal terminal users of that host.

telnet proxy Enables users of one host to log into a remote host and interact as normal terminal users of that host.

3DES Also called triple-DES. Indicates that encryption is performed on a *block* three times with two *keys*: beginning with the first key, then with the second key, and finally with the first key again. See *DES* and *EDE*.

tiff Tagged Image File Format. A format for compressing bitmap files to define how the computer accesses and draws files. See *gif* and *jpeg*.

token Data object or message that describes the current state of the network.

token ring LAN formed in a closed loop topology to regulate online traffic.

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.

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.

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.

UDH Unsigned Diffie-Hellman. 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. See *certificate Identifier (ID)*.

UDP User Datagram Protocol. All CDP communication uses UDP.

unicast *Packet* sent to a single destination. Compare *broadcast*, *multicast*.

UNIX Operating system originally developed at AT&T Bell Laboratories by Ken Thompson and Dennis Ritchie in 1969.

URL Uniform Resource Locator. A code that searches for the location of a specific address on the Internet.

U.S. and Canada use only Version of the SunScreen SKIP certificate software that uses 2048-bit keys and allows users to select 3DES, IDEA, and so forth, for traffic encryption. Compare export controlled and global.

version Manner in which a policy's historical versions are preserved.

user ID Name by which a user is known to the system.

Virtual Private Network A network with the appearance and functionality of a regular network, but which is really like a private network within a public one.

The use of encryption in the lower protocol layers to provide 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.

VPN gateway See *Virtual Private Network*.

VPN See *Virtual Private Network*.

Web See *World Wide Web*.

Web page Document on the Web.

World Wide Web Network of servers on the Internet with one or more home pages that provide information and can include hypertext links to other documents on that server and often other servers as well.

window button Button used to display a *window* containing additional *controls*. See *Button*.

X.509 See *UDH* and *certificate Identifier (ID)*.

Index

NUMERICS

A

B

C

D

E

F

G

H

M

N

O

P

R

S

T

U

V

W

X