Security Products

ScreenOS CLI Reference Guide: IPv6 Command Descriptions

Release 6.1.0, Rev. 01

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089
USA
408-745-2000
www.juniper.net

Part Number: 530-022526-01, Revision 01

Copyright Notice

FCC Statement

The following information is for FCC compliance of Class A devices: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. The equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

The following information is for FCC compliance of Class B devices: The equipment described in this manual generates and may radiate radio-frequency energy. If it is not installed in accordance with Juniper Networks' installation instructions, it may cause interference with radio and television reception. This equipment has been tested and found to comply with the limits for a Class B digital device in accordance with the specifications in part 15 of the FCC rules. These specifications are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Consult the dealer or an experienced radio/TV technician for help.
- Connect the equipment to an outlet on a circuit different from that to which the receiver is connected.

Caution: Changes or modifications to this product could void the user's warranty and authority to operate this device.

Disclaimer

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR JUNIPER NETWORKS REPRESENTATIVE FOR A COPY.

Table of Contents

Chapter 2 exit Through nsmgmt 83

Chapter 3 ntp Through RIPng 175

About This Guide

This guide describes the Internet Protocol version 6 (IPv6) commands used to configure and manage a Juniper Networks security device from a console interface.

NOTE: If a command is not included in this document, it is not supported for this release of ScreenOS.

Document Organization

This guide includes the following sections:

- Chapter 1 describes the CLI commands **active-user** through **event**.
- Chapter 2 describes the CLI commands **exit** through **nsmgmt**.
- Chapter 3 describes the CLI commands **ntp** through **RIPng**.
- Chapter 4 describes the CLI commands **rm** through **zone**.
- Appendix A lists and briefly describes security-device interfaces.
- Appendix B lists and briefly describes zones.

Document Conventions

This document uses the conventions described in the following sections:

- Command Line Interface Conventions on page 8
- Object-Naming Conventions on page 9
- Availability of CLI Commands and Features on page 12

Command Line Interface Conventions

The following conventions are used to present the syntax of command line interface (CLI) commands in examples and in text.

In examples:

- Anything inside square brackets [ ] is optional.
- Anything inside braces { } is required.
- If there is more than one choice, each choice is separated by a pipe (|). For example:

```
set interface { ethernet1 | ethernet2 | ethernet3 } manage
```

- Variables are in *italic* type:

  set admin user *name1* password *xyz*

In text, commands are in **boldface** type and variables are in *italic* type.

NOTE: When entering a keyword, you only have to type enough letters to identify the word uniquely. Typing **set adm u pea j12fmt54** will enter the command **set admin user pearl j12fmt54**. However, all the commands documented here are presented in their entirety.

Each CLI command description in this document reveals some aspect of command syntax. This syntax may include options, switches, parameters, and other features. To illustrate syntax rules, some command descriptions use *dependency delimiters*. Such delimiters indicate which command features are mandatory and in which contexts.

Dependency Delimiters

Each syntax description shows the dependencies between command features by using special characters.

- The { and } symbols denote a mandatory feature. Features enclosed by these symbols are essential for execution of the command.
- The [and] symbols denote an optional feature. Features enclosed by these symbols are not essential for execution of the command, although omitting such features might adversely affect the outcome.
- The | symbol denotes an “or” relationship between two features. When this symbol appears between two features on the same line, you can use either feature (but not both). When this symbol appears at the end of a line, you can use the feature on that line or on the one below it.

Embedded Dependencies

Many CLI commands have *embedded* dependencies, which make features optional in some contexts, and mandatory in others. The two hypothetical features shown below demonstrate this principle.

```
[ feature_1 { feature_2 } ]
```

In this example, the delimiters [and] surround the entire clause. Consequently, you can omit both feature_1 and feature_2, and still execute the command successfully. However, because the delimiters { and } surround feature_2, you must include feature_2 if you include feature_1. Otherwise, you cannot successfully execute the command.

The following example shows some of the feature dependencies of the set interface command.

```
set interface vlan1 broadcast { flood | arp [ trace-route ] }
```

The { and } brackets indicate that specifyng either flood or arp is mandatory. By contrast, the [and] brackets indicate that the trace-route option for arp is not mandatory. Thus, the command might take any of the following forms:

```
device-> set interface vlan1 broadcast flood
device-> set interface vlan1 broadcast arp
device-> set interface vlan1 broadcast arp trace-route
```

Object-Naming Conventions

ScreenOS employs the following conventions regarding the names of objects—such as addresses, admin users, auth servers, IKE gateways, virtual systems, VPN tunnels, and zones—defined in ScreenOS configurations:

- If a name string includes one or more spaces, the entire string must be enclosed within double quotes (“); for example:

  ```
  set address trust “local LAN” 10.1.1.0/24
  ```

- Any leading spaces or trailing text within a set of double quotes are trimmed; for example, “ local LAN ” becomes “local LAN”.

- Multiple consecutive spaces are treated as a single space.

- Name strings are case-sensitive, although many CLI keywords are case-insensitive. For example, “local LAN” is different from “local lan”.

ScreenOS supports the following character types:

- Single-byte character sets (SBCS) and multiple-byte character sets (MBCS). Examples of SBCS are ASCII, European, and Hebrew. Examples of MBCS—also referred to as double-byte character sets (DBCS)—are Chinese, Korean, and Japanese.

- ASCII characters from 32 (0x20 in hexadecimals) to 255 (0xff), except double quotes (“), which have special significance as an indicator of the beginning or end of a name string that includes spaces.

NOTE: A console connection only supports SBCS. The WebUI supports both SBCS and MBCS, depending on the character sets that your browser supports.

Variable Notation

Many CLI commands have changeable parameters, which are presented as variables in Juniper Networks documentation. Variables include names, identification numbers, IP addresses, subnet masks, numbers, dates, and other values.

The variable notation used in this manual consists of italicized parameter identifiers. For example, the set arp command uses four identifiers, as shown here:

```
set arp
    {
    ip_addr mac_addr interface
    age number |
    always-on-dest |
    no-cache
    }
```

where:

- *ip_addr* represents an IP address.
- *mac_addr* represents a MAC address.
- *interface* represents a physical or logical interface.
- *number* represents a numerical value.

Thus, the command might take the following form:

```
device-> set arp 172.16.10.11 00e02c000080 ethernet2
```

where 172.16.10.11 is an IP address, 00e02c000080 is a MAC address, and ethernet2 is a physical interface.

Common CLI Variable Names

Some commands contain multiple variables of the same type. The names of such variables might be numbered to identify each individually. For example, the set dip command contains two *id_num* variables, each numbered for easy identification:

```
set dip group id_num1 [ member id_num2 ]
```

Table 1 lists CLI variable names used in Juniper Networks technical documentation:

Table 1: Command Line Interface Variable Names (Page 1 of 2)

comm_name	The community name of a host or other device
date	A date value
dev_name	A device name, as with flash card memory
dom_name	A domain name, such as “acme” in www.acme.com

Table 1: Command Line Interface Variable Names (Page 2 of 2)

dst_addr	A destination address
filename	The name of a file
fqdn	Fully qualified domain name, such as www.acme.com
grp_name	The name of a group, such as an address group or service group
interface	A physical or logical interface
id_num	An identification number
ip_addr	An IPv4 address or IPv6 address
key_str	A key, such as a session key, a private key, or a public key
key_hex	A key expressed as a hexadecimal number
loc_str	A location of a file or other resource
mac_addr	A MAC address
mbr_name	The name of a member in a group, such as an address group or a service group
mask	A subnet mask, such as 255.255.255.0 or /24
mcst_addr	A multicast address
name_str	The name of an item, such as “Marketing” for an address-book entry
number	A numeric value, usually an integer, such as a threshold or a maximum
pol_num	An integer that identifies a policy
port_num	A number identifying a logical port
pref_len	A number identifying the prefix length for an IPv6 address
pswd_str	A password
ptcl_num	A number uniquely identifying a protocol, such as TCP, IP, or UDP
serv_name	The name of a server
shar_secret	A shared secret value
spi_num	A Security Parameters Index (SPI) number
src_addr	A source address
string	A character string, such as a comment
svc_name	The name of a service, such as HTTP or MAIL
svc_num	The ID number of a service, such as HTTP or MAIL
time	A time value
tunn_str	The name of a tunnel, such as an L2TP tunnel
url_str	A URL, such as www.acme.com
usr_str	A username, usually an external entity such as a dialup user
vrouter	A virtual router, such as trust-vr or untrust-vr
vsys	The name of a vsys (virtual system)
zone	The name of a security zone

Availability of CLI Commands and Features

Some ScreenOS commands are device-specific. Because security devices treat unsupported commands as improper syntax, attempting to execute such a command usually generates the unknown keyword error message. When this message appears, enter the command followed by ? to confirm the availability of the command. For example, the following commands list available options for the set vpn command:

```
device-> set vpn ?
device-> set vpn vpn_name ?
device-> set vpn gateway gate_name ?
```

Filtering

To include or exclude output lines generated by a **get** command, use the piping symbol (|). The general format for such filtering is as follows:

```
get keyword | include string
get keyword | exclude string
```

For example, to filter the output of the get interface command, displaying only lines that contain “eth”, you would enter the following command:

```
get interface | include “eth”
```

To filter the output of the get interface command, displaying no lines that contain “Null”, you would enter the following command:

```
get interface | exclude “Null”
```

Redirection

To direct the output of a get command to a text file on a TFTP server, use the greater-than (>) switch. The general format for such redirection is as follows:

```
get keyword > tftp ip_addr filename
```

For example, to direct the output of the **get address** command to a text file named **addr.txt** on a TFTP server at IP address 172.16.3.4, you would enter the following command:

```
get address > tftp 172.16.3.4 addr.txt
```

The following example shows the WebUI path and parameters for defining an address:

The navigation tree also provides a Help > Config Guide configuration page to help you configure security policies and Internet Protocol Security (IPSec). Select an option from the dropdown menu and follow the instructions on the page. Click the ? character in the upper left for Online Help on the Config Guide.

Technical Documentation and Support

To obtain technical documentation for any Juniper Networks product, visit www.juniper.net/techpubs/.

For technical support, open a support case using the Case Manager link at http://www.juniper.net/customers/support/ or call 1-888-314-JTAC (within the United States) or 1-408-745-9500 (outside the United States).

If you find any errors or omissions in this document, please contact Juniper Networks at techpubs-comments@juniper.net.

Chapter 1

active-user Through event

This section lists and describes ScreenOS command line interface (CLI) commands **active-user** through **event**.

NOTE: Certain commands and features are platform-dependent and might be unavailable on your Juniper Networks security product platform. Check your product datasheet for feature availability.

Click on a command to see details:

active-user
address
admin
alarm
alg
alias
all
arp
attack
auth
auth-server
chassis
common-criteria
config
counter
cpu-limit
delete
dhcp
dip
dns
domain
dot1x
envar
event

active-user

Use the **active-user** command to display information for all users that initiated a service request through the security device. The displayed information includes the IP address of each user, and the number of sessions (incoming and outgoing) currently active for the user.

NOTE: The maximum number of sessions allowed for users depends upon the software license installed on the device.

Syntax

```
get active-user
```

Keywords and Variables

None.

address

Use the **address** commands to define entries in the address book of a security zone.

An *address book* is a list containing all addresses, address groups, and domain names defined for a security zone. You use address-book entries to identify addressable entities in policy definitions.

Syntax

get

```
get address zone [ group [ name_str ] | name name_str ]
```

set

```
set address zone name_str
    {
    fqdn |
    ip_addr/{ mask | pref_len }
    }
      [ string ]
```

Keywords and Variables

Variables

zone The name of the security zone. The default security zones to which you can bind an address book include Trust, Untrust, Global, DMZ, V1-Trust, V1-Untrust, and V1-DMZ. You can also assign address-book entries to user-defined zones.

For more information, see “Interfaces” on page 319.

fqdn The fully qualified domain name of the host.

ip_addr/ { *mask* | *pref_len* } The IPv4 address and subnet mask, or an IPv6 address and prefix length, identifying an individual host or a subnet.

string A character string containing a comment line.

Example1: This example creates an IPv4 address-book entry.

- Entry name of odie
- Address book of the Trust zone
- IP address 10.16.10.1/32
- Assigns the entry a comment string Mary_Desktop

```
set address trust odie 10.16.10.1/32 Mary_Desktop
```

Example 2: This example creates IPv6 address-book entries and uses them to establish incoming and outgoing security policies.

1. Bind physical interfaces to security zones (typically Trust and Untrust), and assign them IP address and network prefixes. (For more information, see “Interfaces” on page 319.)

```
set interface ethernet2 zone trust
set interface ethernet2 ipv6 mode router
set interface ethernet2 ipv6 enable
set interface ethernet2 ipv6 ip 2e80::200:ff:fe00:5/64
set interface ethernet2 ipv6 ra transmit
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode router
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 3ff1::450:af:34ac:77/64
```

2. Create address-book entries (such as NY_Corp_Net and Outside_Net) for the interfaces.

```
set address trust NY_Corp_Net 2e80::200:ff:fe00:0/64
set address untrust Outside_Net 3ff1::450:af:34ac:0/64
```

3. Create security policies that allow tunneled traffic between the peer devices. (For more information, see “policy” on page 195.)

```
set policy from trust to untrust NY_Corp_Net Outside_Net any permit
set policy from untrust to trust Outside_Net NY_Corp_Net http permit
```

group

```
get address zone group name_str
```

group	The name of a group of address-book entries. You can use an address group in a security policy definition to specify multiple addresses.

Example: In the following example you create a group (Our_Group) containing an address (NY_Corp_Net), and display the information in the group:

```
set group address trust Our_Group add NY_Corp_Net
get address trust group Our_Group
```

name

name *name_str*	The name of an individual address-book entry. You can use an address group in a security policy definition to specify a single address.

Example: The following command displays an address named NY_Corp_Net.

```
get address trust name NY_Corp_Net
```

admin

Use the **admin** commands to configure or display administrative parameters for the security device.

These parameters determine the following:

- Characteristics for each admin user, such as password and privilege level
- How the device performs admin user authentication
- Ways that admin users can access the device
- Which IP address to use for administering the device from the Web
- Which port the device uses to detect configuration changes made through the Web
- Whether the device automatically sends email for generated alerts and traffic alarms
- Whether the device is enabled for reset

Syntax

clear

```
clear [ cluster ] admin { all | name name_str }
```

get

```
get admin
    [
    auth [ banner | settings ] |
    current-user |
    manager-ip |
    ssh all |
    user [ trustee | login ]
    ]
```

set

```
set admin
    {
    access attempts number |
    auth
      {
      banner { console login string | secondary string | telnet login string } |
      remote { fallback permit { non-root | root } | primary | read-only | root } |
      server name_str |
      timeout number |
      } |
    device-reset |
    format { dos | unix } |
    http redirect |
    hw-reset |
    mail
      {
      alert |
      mail-addr1 | mail-addr2
        { ip_addr name_str } |
      server-name { ip_addr | name_str } |
      traffic-log
      } |
    manager-ip { ip_addr [ mask ] | enforce } |
    name name_str |
    password [ pswd_str | restrict length number ] |
    port port_num |
    privilege { get-external | read-write } |
    root access console |
    ssh
      {
      password { disable | enable } username name_str |
      port port_num
      } |
    telnet port port_num |
    user name_str
      {
      password pswd_str [ privilege { all | read-only } ] |
      trustee { interface | modem }
      }
```

Keywords and Variables

access attempts

```
set admin access attempts number
unset admin access attempts
```

access attempts
Specifies the number (1 - 255) of unsuccessful login attempts allowed before the device closes the Telnet connection. The default is 3.

Example: The following command sets the number of allowed unsuccessful login attempts to 5:

```
set admin access attempts 5
```

alert

```
set admin mail alert
```

alert
Collects system alarms from the device for sending to an email address.

all

```
clear admin all
```

all
Clears all admin user profiles.

auth

```
get admin auth [ banner | settings ]
set admin auth banner { console | telnet } login string
set admin auth server name_str }
set admin auth timeout number
unset admin auth { banner { console | telnet } login | server | timeout }
```

auth
Configures admin authentication settings for the security device.

- **banner** Specifies the banner (*string*) displayed during login through the console port (**console**) or a Telnet session (**telnet**). You can also specify a secondary banner.
- **server** The name of the authentication server used for authenticating admin users.
- **timeout** Specifies the length of idle time (in minutes) before the security device automatically closes the Web administrative session. The value can be up 999 minutes. A value of 0 specifies no timeout. (Telnet admin sessions time out after the console timeout interval expires. You set this interval using the **set console timeout** command.)

Example2: The following commands create two login banners:

- “HyperTerminal Management Console”—displayed at the start of new console admin sessions
- “Telnet Login Here”—displayed at the start of new Telnet admin sessions

```
set admin auth banner console login "HyperTerminal Management Console"
set admin auth banner telnet login "Telnet Login Here"
```

cache

clear [cluster] admin user cache
get admin user cache

cache — Clears or displays the memory cache containing all current remote administrative users.

cluster

clear cluster admin user { cache | login }

cluster — Propagates the clear operation to all other devices in an NSRP cluster.

Example: The following command clears remote administrative users from the cache and propagates this change to other devices in an NSRP cluster:

```
clear cluster admin user cache
```

current-user

get admin current-user

current-user — Displays the user for the current administrative session.

device-reset

set admin device-reset
unset admin device-reset

device-reset — Enables device reset for asset recovery.

format

set admin format { dos | unix }
unset admin format

format — Determines the format (dos or unix) used when the security device generates the configuration file. On some Juniper Networks platforms, you can download this file to a TFTP server or PCMCIA card using the CLI or to a local directory using the WebUI.

http redirect

set admin http redirect
unset admin http redirect

http redirect	Allows you to manage and redirect HTTP traffic.

hw-reset

set admin hw-reset
unset admin hw-reset

hw-reset	Enables hardware reset for asset recovery.

login

clear [cluster] admin user login
get admin user login

login	Clears or displays all current administrative users.

mail

set admin mail { ... }
unset admin mail { ... }

mail	Enables email for sending alerts and traffic logs.

Example: The following command configures the email address *john@abc.com* to receive updates concerning administrative issues:

set admin mail mail-addr1 john@abc.com

mail-addr1

set admin mail mail-addr1 { *ip_addr* **|** *name_str* }

mail-addr1	Sets the first email address (such as chris@acme.com) for sending alert and traffic logs.

mail-addr2

set admin mail mail-addr2 { *ip_addr* **|** *name_str* }

mail-addr2	Sets the secondary email address for sending alert and traffic logs.

Example: The following command configures the secondary email address *pat@acme.com* to receive updates concerning administrative issues:

set admin mail mail-addr2 pat@acme.com

manager-ip

```
get admin manager-ip [ all device ]
set admin manager-ip { ip_addr [ mask ] | enforce }
unset admin manager-ip { ip_addr | all [ device ] | enforce }
```

manager-ip

Restricts management to a host or a subnet. The default **manager-ip** address is 0.0.0.0, which allows management from any workstation. The number of manager IP addresses you can set depends on the security device. On devices that support Vsys, the number of manager IP addresses is 50 plus one times the number of Vsys.

- **all** Displays or removes all manager IP addresses for the current Vsys.
- **device** Displays or removes all manager IP addresses from all Vsys on the device.manager-ip
- **enforce** Specifies that all Vsys be configured to use **manager-ip**.
- **ip_addr/mask** Adds the specified IP address to the list of manager IP addresses for the current Vsys.

Note: The **manager-ip** address must be unique, and different from the physical IP address of the management interface.

name

```
set admin name name_str
unset admin name
```

name

The login name (*name_str*) of the root user for the security device. The maximum length of the name is 31 characters, including all symbols except ?. The name is case-sensitive.

password

```
set admin password pswd_str
unset admin password
```

password

Specifies the password (*pswd_str*) of the root user. The maximum length of the password is 31 characters, including all symbols except the special command character ?.

port

```
set admin port port_num
unset admin port
```

port

Sets the port number (*port_num*) for detecting configuration changes when using the Web. Use any number between 1024 and 32767, or use the default port number (80). Changing the admin port number might require resetting the device (see the **reset** command).

privilege

```
set admin privilege ( get-external | read-write }
```

privilege — Defines the administrative privilege level:

- **get-external** Instructs the security device to obtain the admin user privileges externally from the RADIUS server.
- **read-write** Gives the RADIUS administrator read-write privileges, and ignores the privilege returned from the RADIUS server.

restrict length

```
set admin password restrict length number
unset admin password restrict length
```

restrict length — Sets the minimum password length of the root admin. The password length can be any number from 1 to 31.

root access console

```
set admin root access console
unset admin root access console
```

root access console — Restricts the root admin to logging in to the device through the console only.

server-name

```
set admin mail server-name ip_addr | name_str
```

server-name — The IP address or name of the Simple Mail Transfer Protocol (SMTP) server. This server receives email notification of system alarms and traffic logs.

Example: The following command specifies a SMTP server at IP address 10.1.10.10:

```
set admin mail server-name 10.1.10.10
```

settings

```
get admin auth settings
```

settings — Displays admin authentication settings, including the current timeout setting and the admin user type (local or remote).

ssh

```
get admin ssh all
set admin ssh password { disable | enable } username name_str
set admin ssh password port port_num
unset admin ssh [ port ]
```

ssh

Provides access to the Secure Shell (SSH) utility. SSH allows you to administer security devices from an Ethernet connection or a dial-in modem, thus providing secure CLI access over unsecured channels.

- all Displays the SSH PKA (Public Key Authentication) information for each admin.
- password Sets the password for the user that establishes the SSH session. The enable | disable switch enables or disables password authentication. username *name_str* specifies the admin username.
- port *port_num* Specifies the logical SSH port through which the communication occurs. The default is port 22. Unsetting the port resets the SSH port to the default.

telnet

```
set admin telnet port port_num
unset admin telnet port
```

telnet port

Provides CLI access through a Telnet connection. The acceptable range of *port_num* is 1024 - 32767.

traffic-log

```
set admin mail traffic-log
unset admin mail traffic-log
```

traffic-log

Generates a log of network traffic handled by the security device. The traffic log can contain a maximum of 4,096 entries. The security device sends a copy of the log file to each specified email address (see mail-addr1 and mail-addr2). This happens when the log is full, or every 24 hours, depending upon which occurs first.

user

```
get admin user [ cache | login ]
set admin user name_str password pswd_str [ privilege { all | read-only } ]
unset admin user name_str
```

user

Creates or displays a non-root administrator (super-administrator or sub-administrator). The maximum username length is 31 characters, including all symbols except ?. The username is case-sensitive. The privilege switch determines the privilege level of the user (all or read-only).

Example: The following command creates a non-root administrator named rsmith with password swordfish:

```
set admin user rsmith password swordfish privilege all
```

Defaults

- The default admin name and password are netscreen.
- The default number of access attempts is 3.
- The default manager-ip is 0.0.0.0, and the default subnet mask is 255.255.255.255.
- The default privilege for a super-administrator is read-only.
- The default admin port is 80.
- The default mail alert setting is off.
- The default for device reset is on.

alarm

Use the alarm commands to set or display alarm parameters.

The alarm parameters determine when the device generates alarm messages, and the amount and type of information contained in the messages.

Syntax

clear

```
clear [ cluster ] alarm traffic
    [ policy pol_num1 [ -pol_num2 ] ]
      [ end-time string ]
```

get

```
get alarm
    {
    snapshot cpu { alarm_time string | all } |
    threshold |
    traffic
      [ policy { pol_num1 [ -pol_num2 ] } ]
        [ service name_str ]
          [ src-address ip_addr ] [ dst-address ip_addr ]
            [ detail
              [ start-time string ] [ end-time string ]
                [ minute | second
                  [ threshold number [ -number ] ]
                    [ rate number [ -number ] ]
                ]
            ] |
    }
```

set

```
set alarm threshold
    {
    cpu number |
    memory number |
    session { count number | percent number }
    }
```

Keywords and Variables

cluster

clear cluster alarm traffic [...]

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

Example: The following command clears the alarm table entries for policy 4 and propagates the change to other device in an NSRP cluster:

clear cluster alarm traffic policy 4

detail

get alarm traffic [...] detail [...]

detail	Displays detailed information for each policy, including all traffic alarm entries that occurred under the policy. If you omit this option, the output contains only general information and the time of the most recent alarm for each policy.

Example: The following command displays event alarm entries or traffic alarm entries that occur on or after January 1, 2003:

get alarm traffic detail start-time 01/01/2003

end-time | start-time

clear [cluster] alarm traffic policy [...] end-time *number*
get alarm traffic [...] end-time *number*
get alarm traffic [...] start-time *number*

start-time end-time	The start-time option displays event alarm entries or traffic alarm entries that occurred at or before the time specified. The end-time option displays event alarm entries or traffic alarm entries that occurred at or after the time specified. The format for *string* is *mm*/*dd*[/*yy*-*hh*:*mm*:*ss*] You can omit the year (the current year is the default), or express the year using the last two digits or all four digits. The hour, minute, and second are optional. The delimiter between the date and the time can be a dash or an underscore: 12/31/2002-23:59:00 12/31/2002_23:59:00

Example: The following command performs a detailed display of traffic alarm entries at (or after) 11:59pm, December 31, 2003 and at or before 12:00am, December 31, 2004:

```
get alarm traffic detail start-time 12/31/2003-23:59:00 end-time
    12/31/2004-24:00:00
```

policy

clear [cluster] alarm traffic policy *pol_num1* [-*pol_num2*] [...]
get alarm traffic policy *pol_num*

policy

Displays traffic alarm entries for a policy specified by its ID number or for several policies specified by a range of ID numbers. The ID number can be any value between 0 and the total number of established policies. To define a range, enter the starting and ending ID numbers as follows: *pol_num1-pol_num2*

Example: The following command clears the entries for policy 2 in the alarm table:

```
clear alarm traffic policy 2
```

second | minute

get alarm traffic [...] detail

second | minute

Displays traffic alarm entries for policies with threshold settings at bytes per second or bytes per minute.

- The rate *number* [-*number*] option displays traffic alarm entries for policies with a flow rate at a specified value or within a specified range.
- The threshold *number* [-*number*] option displays traffic alarm entries for policies with a threshold at a specified value or within a specified range.

Example: The following command displays traffic alarm entries for policies with threshold settings at bytes per second:

```
get alarm traffic detail second
```

service

get alarm traffic [...] service *name_str* [...]

service

Displays traffic alarm entries for a specified service (*name_str*), such as TCP, ICMP, or FTP. (To display all services, make the *name_str* value Any.) The name does not have to be complete; for example, both TC and CP are recognized as TCP. Although you cannot specify a Service group, note that because TP is recognized as FTP, HTTP, and TFTP, entering TP displays traffic-alarm entries for all three of these services.

Example: The following command displays traffic alarm entries for the HTTP service:

```
get alarm traffic service http
```

snapshot

get alarm snapshot cpu { alarm_time *string*| all }

snapshot	Displays snapshots triggered by a CPU alarm. ■ alarm_time *MM/DD/YYYY-hh:mm:ss* shows a snapshot of a specific time that you request. ■ all shows all snapshots.

src-address | dst-addr

get alarm traffic [...] src-address *ip_addr* [...]
get alarm traffic [...] dst-address *ip_addr* [...]

src-address	Displays traffic alarm entries originating from a specified IP address (*ip_addr*) or from a specified direction, such as inside_any or outside_any.
dst-address	Displays traffic alarm entries destined for a specified IP address (*ip_addr*) or for a specified direction, such as inside_any or outside_any.

Example 1: The following command displays traffic alarm entries originating from IP address 10.1.9.9 and destined for IP address 1.1.10.10:

get alarm traffic src-address 10.1.9.9 dst-address 1.1.10.10

Example 2: The following command displays traffic alarm entries originating from IPv6 address 2001::5 and destined for IPv6 address 2001::25:

get alarm traffic src-address 2001::5 dst-address 2001::25

threshold

get alarm threshold
get alarm traffic [...] threshold *number* [*-number*]
set alarm threshold { ... }
unset alarm threshold { CPU | memory | session }

threshold	Displays traffic alarm entries for policies with threshold settings at a specified value or within a specified range. ■ cpu *number* sets the cpu threshold. ■ memory *number* sets the memory threshold. ■ session sets the session threshold. The count *number* option specifies how many sessions can exist before the device generates an alarm. The percent *number* option specifies what percentage of the session limit is allowable before the device generates an alarm.

Example: The following command sets the session limit threshold to 75,000 sessions:

set alarm threshold session count 75000

traffic

```
clear [ cluster ] alarm traffic [ ... ]
get alarm traffic [ ... ]
```

traffic	Specifies traffic alarm entries.

Example: The following command performs a detailed display of traffic alarm entries originating from IP address 10.1.9.9 and destined for IP address 1.1.10.10:

get alarm traffic src-address 10.1.9.9 dst-address 1.1.10.10 detail

alg

Use the alg commands to enable or disable an Application Layer Gateway (ALG) on the security device. An ALG runs as a service and can be associated in policies with specified types of traffic. ALGs are enabled by default.

Syntax

clear

```
clear alg { mgcp counters }
```

get

```
get alg
    [
    h323 |
    mgcp
      [
      calls |
      counters |
      endpoints [ name string ] |
      sessions [ dst-ip ip_addr | src-ip [ ip_addr ] ]
      ] |
    msrpc |
    pptp |
    rtsp |
    sip { call | setting } |
    sql |
    sunrpc
    ]
```

set

```
set alg
    {
    h323
      {
      app-screen unknown-message [ nat | route ] permit |
      enable |
      gate source-port-any |
      incoming-table timeout number
      { |
    mgcp
      {
      app-screen
        {
        connection-flood [ threshold number ] |
        message-flood [ threshold number ] |
        unknown-message [ nat | route ] permit
        } |
      enable |
      inactive-media-timeout number  |
      max-call-duration number |
      transaction-timeout number
      } |
    msrpc [ enable ] |
    pptp [ enable ] |
    rtsp [ enable ] |
    sip
      {
      app-screen
       {
        protect deny [ dst-ip [ipv4-addr / netmask | ipv6 ipv6-addr /prefix_length] |
        timeout number ] | unknown-message [ nat | route] permit ]
        } |
        enable |
        media-inactivity-timeout number |
        signaling-inactivity-timeout number
      } |
    sql [ enable ] |
    sunrpc [ enable ]
    }
```

Keywords and Variables

h323

```
get alg h323 [ ... ]
set alg h323 [ ... ]
unset alg h323 [ ... ]
```

h323

Specifies the H.323 ALG on the device. H.323 is a control signaling protocol used to exchange messages between H.323 endpoints.

- **app-screen unknown-message [nat | route] permit** specifies how unidentified messages are handled. The default is to drop unknown messages.
  - **nat** specifies that unknown messages be allowed to pass even if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass even if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **enable** enables and disables the H.323 ALG (the default is enabled).
- **gate source-port-any** specifies that the security device accept calls from any port number.
- **incoming-table timeout** specifies the timeout value in seconds for entries in the NAT table. The default is 3600 seconds.

mgcp

```
get alg mgcp [ ... ]
set alg mgcp [ ... ]
unset alg mgcp [ ... ]
clear alg mgcp counters
```

mgcp

Specifies the MGCP ALG on the device. MGCP is a text-based application layer protocol that can be used for call setup and call control.

- **app-screen connection-flood threshold** specifies the threshold for connections per second, limiting the rate of processing CreateConnection requests from the call agent and thereby constraining pinhole creation. CreateConnection requests that exceed this threshold are dropped. Disabled by default. When enabled, default threshold value is 200 connections; minimum is 10, maximum is 1000.
- **app-screen message-flood threshold** specifies the rate in seconds beyond which messages arriving on an MGCP session are dropped. Disabled by default. When enabled, default is 1000 messages; minimum is 50, maximum is 500.
- **app-screen unknown-message [nat | route] permit** specifies how unidentified messages are handled. The default is to drop unknown messages.
  - **nat** specifies that unknown messages be allowed to pass even if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass even if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **calls** displays active MGCP calls.
- **counters** displays MGCP statistics.

- **enable** enables and disables the MGCP ALG (the default is enabled).
- **endpoints** displays endpoints of active sessions.
- **inactive-media-timeout** specifies how long pinholes and sessions opened for media are kept alive in the absence of activity. The default is 120 seconds; minimum is 10 seconds, maximum is 2550 seconds.
- **max-call-duration** specifies the maximum number of minutes (the default is 720) extablished calls are kept alive. The minimum is 3, maximum is 1440.
- **sessions** displays MGCP session information.
  - **dst-ip** matches the destination IP address of the session.
  - **src-ip** matches the source IP address of the session.

msrpc

```
get alg msrpc
set alg msrpc enable
unset alg msrpc enable
```

msrpc	Specifies the Microsoft Remote Procedure Call ALG on the device (the default is enabled).

pptp

```
get alg pptp
set alg pptp enable
unset alg pptp enable
```

pptp	Specifies the Point to Point Tunneling Protocol on the device (the default is enabled).

rtsp

```
get alg rtsp
set alg rtsp enable
unset alg rtsp enable
```

rtsp

Specifies the Real Time Streaming Protocol ALG on the device (the default is enabled).

Note: In a dual stack environment, the **set** command will enable both IPv4 and IPv6 RTSP ALGs at the same time. Similarly, the unset command will disable the IPV4 and IPV6 RTSP ALGs. The limitations of this feature are:

- ALG does not support NAT for IPv6.
- ALG does not support transparent mode for IPv6.
- ALG does not support NetScreen Redundancy Protocol (NSRP) for IPv6.

sip

```
get alg sip [ … ]
set alg sip [ … ]
unset alg sip [ … ]
```

sip

Specifies the Session Initiation Protocol ALG on the device.

- **app-screen unknown-message** specifies how unidentified messages are handled. The default is to drop unknown messages.
  - **nat** specifies that unknown messages be allowed to pass even if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass even if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **call** Displays the number of active calls. The maximum number of calls possible on a security device depends on the platform type. For more information, refer to the datasheet for your security device.
- **enable** enables and disables the SIP ALG on the device (the default is enabled).
- **media-inactivity-timeout** specifies how long pinholes and sessions opened for media are kept alive in the absence of activity. The default is 120 seconds; minimum is 10 seconds, maximum is 2550 seconds.
- **protect deny** specifies that repeat SIP INVITE requests be denied to a proxy server that denied the initial request.
  - **dst-ip** specifies the IP address of the proxy server.
  - **ipv6** specifies the IPv6 address of the proxy server.
  - **timeout** specifies the time in seconds the proxy server denies repeated SIP messages before it begins accepting them again. The default is 5 seconds; the range is 1 to 3600 seconds.
- **setting** displays the inactivity timeout parameters for SIP signaling and media, and the destination address of a SIP proxy server protected from repeat SIP INVITE requests from the proxy server initially rejected.
- **signaling-inactivity-timeout** Configures or removes the maximum length of time in seconds a call can remain active without any SIP signaling traffic. Each time a SIP signaling message occurs within a call, this timeout resets. The default setting is 43200 seconds (12 hours); minimum is 10, maximum is 65535.

sql

```
get alg sql
set alg sql enable
unset alg sql enable
```

sql

Specifies the SQL ALG on the device (the default is enabled).

sunrpc

```
get alg sunrpc
set alg sunrpc enable
unset alg sunrpc enable
```

sunrpc

Specifies the Sun Remote Procedure Call ALG on the device (the default is enabled).

alias

Use the alias commands to create, remove, or list aliases. An *alias* is a named variable containing the initial characters of a CLI command. After creating an alias, you can use it to execute the represented command.

Syntax

get

get alias

set

set alias *name_str string*

Keywords and Variables

Variables

name_str	The name of the CLI command alias.
string	The CLI command to which you assign the alias.

Example: The following commands create an alias representing the get interface ethernet1/1 command, then execute the command using the alias:

```
set alias int_1 "get interface ethernet1/1"
int_1
```

all

Use the all command to return all configuration settings to the factory default values.

Syntax

unset all

Keywords and Variables

None.

Example

In the following example, you reset the device to its factory default settings and reset the device.

1. Execute the unset all command.

   unset all

   The following prompt appears: “Erase all system config, are you sure y / [n]?”

2. Press the Y key. This action returns the system configuration to the factory default settings.

3. Execute the reset command.

   reset

   The following prompt appears: “Configuration modified, save? [y] / n”

4. Press the N key. This action generates the following prompt: “System reset, are you sure? y / [n] n”

5. Press the Y key. This action restarts the system. The device now has its original factory default settings.

arp

Use the arp commands to create, remove, or list interface entries in the Address Resolution Protocol (ARP) table of the security device.

Syntax

clear

```
clear [ cluster ] arp [ ip_addr | all ]
```

get

```
get arp [ all | asic id_num ]
```

set

```
set arp
    {
    ip_addr mac_addr interface |
    age number |
    always-on-dest
    }
```

Keywords and Variables

Variables

set arp *ip_addr mac_addr interface*

ip_addr	The IP address for the interface in the ARP table entry.
mac_addr	The MAC address for the interface in the ARP table entry.
interface	The name of the ARP interface in the ARP table entry. For more information, see “Interfaces” on page 319.

all

get arp all

age	Lists all current ARP entries for every existing vsys.

asic

get asic *id_num*

asic	Lists all current ARP entries for every a specified ASIC chip (*id_num*).

age

set arp age *number*

age	Sets the age-out value (in seconds) for ARP entries.

always-on-dest

set arp always-on-dest

always-on-dest	Directs the security device to send an ARP request for any incoming packet with a heading containing a MAC address not yet listed in the device’s MAC address table. This may be necessary when packets originate from server load-balancing (SLB) switches or from devices using the Hot Standby Router Protocol/Virtual Router Redundancy Protocol (HSRP/VRRP).

cluster

clear [cluster] arp

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

attack

Use the attack commands to view and define attack objects, attack object groups, and the attack object database server settings.

NOTE: This command is available only if Advanced mode is installed on the device.

Syntax

get

```
get attack
    [
    name_str |
    anomaly [ sort-by { id | name } ] |
    db |
    disable [ sort-by { def-type | id | name | type } ] |
    group [ name_str | sort-by { def-type | name } ] |
    id id_num |
    signature sort-by { def-type | id | name } |
    sort-by { def-type | id | name | type }
    ]
```

set

```
set attack
    {
    CS:name_str
      {
      aim-chat-room-desc |
      aim-chat-room-name |
      aim-get-file |
      aim-nick-name |
      aim-put-file |
      aim-screen-name |
      dns-cname |
      ftp-command |
      ftp-password |
      ftp-pathname |
      ftp-username |
      gnutella-http-get-filename |
      http-authorization |
      http-header-user-agent |
      http-request |
      http-status |
      http-text-html |
      http-url |
      http-url-parsed |
      http-url-variable-parsed |
      imap-authenticate |
      imap-login |
      imap-mailbox |
      imap-user |
      msn-display-name |
      msn-get-file |
   msn-put-file |
   msn-sign-in-name |
   pop3-auth |
   pop3-header-from |
   pop3-header-line |
   pop3-header-subject |
   pop3-header-to |
   pop3-mime-content-filename |
   pop3-user |
   smb-account-name |
   smb-connect-path |
   smb-connect-service |
   smb-copy-filename |
   smb-delete-filename |
   smb-open-filename |
   smtp-from |
   smtp-header-from |
   smtp-header-line
   smtp-header-subject
   smtp-header-to |
   smtp-mime-content-filename |
   smtp-rcpt |
   stream256 |
   ymsg-alias |
   ymsg-chatroom-message |
   ymsg-chatroom-name |
   ymsg-nickname |
   ymsg-p2p-get-filename-url |
   ymsg-p2p-put-filename-url |
   ymsg-user-name
   }
      [ not ] string
         severity { info | low | medium | high | critical } |
db
   {
   mode { notification | update } |
   schedule
      {
      daily hh:mm |
      monthly number hh:mm |
      weekly day hh:mm
      } |
   server url_str
   sigpack { base | client | server | worm }
   } |
disable name_str |
group name_str1 [ add name_str2 ] |
}
```

Keywords and Variables

Variables

```
get attack name_str
set attack name_str aim-chat-room-desc string severity string
…
set attack name_str ymsg-user-name string severity string
unset attack name_str
```

name_str Defines the attack-object name. If it is a user-defined attack, it must be prefaced with *CS:*.

Specifies one of the following contexts for Deep Inspection (DI) to search and defines the signature string for which the DI module searches:

- aim-chat-room-desc *string*

  …

- ymsg-user-name *string*

severity Defines the severity level of the attack. You can specify any of the following levels: info, low, medium, high, critical.

NOTE: For a complete list of contexts that you can specify when creating your own attack objects, refer to *Volume 4: Attack Detection and Defense Mechanisms* in the *Concepts & Examples ScreenOS Reference Guide.*

Example: The following command creates an attack object for FTP named “CS:rootuser”, specifies its context as “ftp-username”, defines its signature as “root”, and specifies its severity level as “high”:

set attack CS:rootuser ftp-username root severity high

anomaly

```
get attack anomaly [ sort-by { id | name } ]
```

anomaly Specifies the attack object as an anomaly.

sort-by Indicates the organization for the display of protocol anomalies in the local database—either numerically by id or alphabetically by name.

attack

```
get attack
```

attack Displays all attack objects currently stored in the local database, displaying—in alphabetical order—first user-defined attacks (if any) and then predefined attacks.

db

get attack db
set attack db mode { notification **|** update }
set attack db schedule { daily *hh:mm* **|** monthly number *hh:mm* **|** weekly day *hh:mm* }
set attack db server *url_str*
unset attack db { mode **|** schedule **|** server **|** sigpack }

db

Specifies the attack-object database server. On security devices that support virtual systems, you must set this command at the root level.

mode Selects either **notification** or **update** as the mode for checking and updating the attack-object database. The **notification** method automatically checks the attack-object database server at user-defined times and notifies the admin if the database on the server is more recent than the one on the security device. (If the data on the server is more recent, a notice appears on the WebUI main page and in the CLI after you log into the device.) The **update** method automatically checks the attack object database server at user-defined times and automatically updates the database on the security device if it determines that the database on the server is more recent than the one on the security device.

Unsetting this command stops the security device from automatically checking the server.

schedule *string* Sets the time for automatically checking the attack-object database server and updating the attack object database on the security device. You can set a daily, monthly, or weekly schedule.

server *url_str* Defines the URL of the attack-object database server. ScreenOS provides four predefined DI signature packs: base, server, client, and worm. The base signature pack is the default. If you do not specify a signature pack as shown in Example 1, then the basic signature pack is retrieved.

Unsetting the attack object database server retrieves the basic signature pack only. If you run the **exec attack-db update** command with a server URL set to null, then the base signature pack from the following URL is loaded: https//services.netscreen.com/restricted/sigupdates

sigpack Specifies the predefined signature packs. To use a signature pack, you must purchase a DI database license key and download the appropriate package for your environment from the Juniper Networks website.

ScreenOS provides four predefined DI signature packs:

- **base** Includes a sample of worm, client-to-server, and server-to-client signatures for Internet-facing protocols and services, such as HTTP, DNS, FTP, SMTP, POP3, IMAP, NetBIOS/SMB, MS-RPC, P2P, and IM (AIM, YMSG, MSN, and IRC).
- **server** Focuses on protecting a server farm. It includes a comprehensive set of server-oriented protocols, such as HTTP, DNS, FTP, SMTP, IMAP, MS-SQL, and LDAP. Also includes worm signatures that target servers.
- **client** Focuses on protecting users from getting malware, Trojans, and so on while surfing the Internet. Includes a comprehensive set of client-oriented protocols, such as HTTP, DNS, FTP, IMAP, POP3, P2P, and IM (AIM, YMSG, MSN, and IRC). Also includes worm signatures that target clients.
- **worm** Includes stream signatures and primarily focuses on providing comprehensive worm protection. Detects server-to-client and client-to-server worm attacks for all protocols.

The base signature pack is the default. If you do not specify a signature pack as shown in Example 1, then the base signature pack is retrieved.

Note: Your security device allows you to load one signature pack at a time.

The **unset attack db sigpack** command followed by the **exec attack-db update** command retrieves the basic signature pack.

Example 1: The following command configures your security device to retrieve the server signature pack:

```
set attack db sigpack server
```

Example 2: Use the following URL strings to configure your security device to retrieve the base, server, client, or worm signature packs, respectively:

```
set attack db server http://services.netscreen.com/restricted/sigupdates
set attack db server http://services.netscreen.com/restricted/sigupdates/server
set attack db server http://services.netscreen.com/restricted/sigupdates/client
set attack db server http://services.netscreen.com/restricted/sigupdates/worm
```

Example 3: The following commands define the URL of the attack-object database server and set a schedule to check the server automatically and then notify the security device admin when the database on the server is more recent than that on the security device:

```
set attack db server http://www.juniper.net/attacks
set attack db schedule daily 07:00
set attack db mode notification
```

disable

set attack disable *name_str*
unset attack disable *name_str*
get attack disable
get attack disable sort-by [sort-by { def-type | id | name | type }]

disable

Disables the specified predefined attack object or a list of all disabled attack objects. You can organize the display of the list by one of the following attributes:

- **def-type**: Organizes the disabled attack-object display by anomaly and then by signature, and then within each of these two categories, alphabetically by protocol.
- **id**: Organizes the disabled attack-object display numerically by ID number.
- **name**: Organizes the disabled attack-object display alphabetically by attack name.
- **type**: Organizes the disabled attack-object display alphabetically by anomaly and then by signature.

group

get attack group [sort-by { def-type | name }]
set attack group *name_str1* [add *name_str2*]
unset attack group *name_str1* [remove *name_str2*]

group

Specifies an attack object group.

sort-by

Indicates the organization for the display of attack groups from the local database:

- **def-type**: Organizes the attack group display by the definition type of the attack group, displaying first the predefined attack groups in alphabetical order and then the user-defined groups.
- **name**: Organizes the attack group display alphabetically by attack group names.

name_str specifies a name for the creation, deletion, or modification of an attack group. The keywords **add** and **remove** indicate that an attack is being added to or is being removed from the specified group.

Example: The following command displays all the attack groups on the security device by name in alphabetical order:

get attack group sort-by name

id

get attack id *id_num*

id

Specifies the ID number of an attack object in the local database.

Example: The following command displays the attack object with ID number 500 in the security device:

get attack id 500

not

set attack CS:*name_str* not string1 severity *string2*

not	Defines as an attack object anything in the specified context except the user-defined attack pattern.

Example: The following command defines the attack object named CS:badlogin as anything except the permitted FTP username “jj2345” with a medium-level severity:

```
set attack CS:badlogin ftp-username not jj2345 severity medium
```

signature

get attack signature [sort-by { def-type | id | name }]

signature

Displays stateful signature attack objects currently stored in the local database.

- sort-by: Specifies the organizational display of attack objects by one of the following attributes:
  - def-type: Organizes the stateful signature attack object display by the definition type of the attack object, displaying first the predefined attack objects in alphabetical order and then the user-defined objects.
  - id: Organizes the stateful signature attack object display numerically by ID number.
  - name: Organizes the stateful signature attack object display alphabetically by attack name.

Example: The following command displays signature attack objects on the security device alphabetically by name:

```
get attack signature sort-by name
```

sort-by

get attack sort-by { def-type | id | name | type }

sort-by

Specifies the organizational display of attack objects in the local database by one of the following attributes:

- def-type: Organizes the attack object display by the definition type of the attack object, displaying first the predefined attack objects in alphabetical order and then the user-defined objects.
- id: Organizes the attack object display numerically by ID number.
- name: Organizes the attack object display alphabetically by attack name.
- type: Organizes the attack object display alphabetically by protocol type, and then within each protocol type by attack name.

Example: The following command displays all attack objects in the security device organized numerically:

```
get attack sort-by id
```

auth

Use the **auth** commands to specify a user authentication method.

The four available methods include:

- A built-in database
- A RADIUS server
- SecurID
- Lightweight Directory Access Protocol (LDAP)
- Terminal Access Controller Access Control System+ (TACACS+)

NOTE: If the security device uses SecurID to authenticate users, and communication problems occur with the ACE server, clear the current SecurID shared secret from the device (and the server) by executing the **clear node_secret** command.

Syntax

clear

```
clear [ cluster ] auth
    [
    history |
    queue |
    statistics |
    table [ id id_num | ip ip_addr ]
    ]
```

get

```
get auth
    [
    banner |
    history [ id id_num | ip ip_addr ] |
    queue |
    settings |
    table [ id id_num | ip ip_addr ]
    ]
```

set

```
set auth
    {
    banner { ftp | http | telnet } { fail string | login string | success string }
    default auth server name_str |
    radius accounting
      {
      action cleanup-session |
      port port_num
      } |
    }
```

Arguments

banner

get auth banner
set auth banner { ftp | http | telnet }
unset auth banner { ftp | http | telnet }

banner	Defines or displays firewall banners. The security device uses these banners to report success or failure of login requests. ■ ftp Reports on the success or failure of FTP login requests. ■ http Reports on the success or failure of HTTP login requests. ■ telnet Reports on the success or failure of Telnet login requests. ■ fail *string* Specifies a message string to display a login attempt fails. ■ login *string* Specifies a message string to display when a login attempt occurs. ■ success *string* Specifies a message string to display when a login attempt is successful.

Example: The following command defines a banner for a failed FTP login attempt:

set auth banner ftp fail "FTP login attempt failed"

cluster

clear [cluster] auth [...]

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

default

set auth default auth server *name_str*
unset auth default auth server

default auth server	Specifies a default firewall authentication server (*name_str*). The security device uses this server when a security policy does not explicitly identify a particular authentication server.

Example: The following command identifies the default authentication server (Auth_Server):

set auth default auth server Auth_Server

history

clear [cluster] auth history
get auth history [id *id_num* | ip *ip_addr*]

history	Clears or displays the history of users authenticated through the security device.

queue

```
clear [ cluster ] auth queue
get auth queue
```

queue — Clears or displays the internal user authentication queue.

radius accounting

```
set auth radius accounting action cleanup-session
set auth radius accounting port port_num
unset auth radius accounting action cleanup-session
unset auth radius accounting port
```

radius accounting — This feature allows any organization that owns or controls a RADIUS server to track RADIUS session information for billing, monitoring, or other purposes. For example, a RADIUS server might need to record information about when authorized sessions begin, when they end, the number of bytes or packets exchanged during each session, and so on. Such tracking is generally referred to as *RADIUS accounting*. Each RADIUS accounting session begins when the RADIUS server receives an Accounting-Start message and ends when it receives an Accounting-Stop message.

RADIUS accounting allows the device to monitor and manage authorized sessions. For example, a device might clear out zombie sessions when it receives an Accounting-Stop message from an external RADIUS client. This could prevent misuse of wireless calls if a subsequent user gets a previous user's assigned IP address and attempts to use the previous user's session.

The port (*port_num*) setting specifies the port through which the device receives Accounting-Start and Accounting-Stop messages. In addition, the **cleanup-session** feature allows the device to clear out zombie sessions when it receives an Accounting-Stop message from an external RADIUS client. This feature prevents misuse of wireless calls if subsequent users get the same assigned IP address and happen to use the previous user's session in the device.

Note: This feature is not supported on the vsys level and is for the root level only.

Note: This feature is only for clearing zombie sessions. Enablement is not required for the security device to support RADIUS accounting while communicating with the RADIUS server.

settings

```
get auth settings
```

settings — Displays default user authentication server settings. (This option yields the same display as the **get auth** command.

statistics

```
clear auth statistics
get auth statistics
```

statistics	Clears or displays authentication run-time statistics.

table

```
clear [ cluster ] auth table [ id id_num | ip ip_addr ]
get auth table [ id id_num | ip ip_addr ]
```

table

Clears entries from the user authentication table (thus forcing reauthentication), or displays such entries. Entries in the user authentication table can represent:

- Users currently authenticated
- Users currently undergoing authentication
- Users denied authentication

Without parameters (described below), the table option clears or displays all table entries.

- id *id_num* Clears or displays a particular entry by ID (*id_num*).
- ip *ip_addr* Clears or displays all entries with a common source IP address (*ip_addr*).

Examples: The following command clears entry 7 from the user authentication table:

clear auth table id 7

The following command displays authentication details from a table entry with source IP 10.1.10.10:

get auth table ip 10.1.10.10

auth-server

Use the auth-server commands to configure the security device for user authentication with a specified authentication server. Admins, policies, VPN tunnel specifications, and XAuth configurations use these server specifications to gain access to the appropriate resources.

Syntax

get

```
get auth-server
    {
    string |
    all |
    id id_num
    }
```

set

```
set auth-server name_str
    {
    account-type { [ 802.1X ] [ admin ] | [ auth ] [ l2tp ] [ xauth ] } |
    backup1 { ip_addr | name_str } |
    backup2 { ip_addr | name_str } |
    fail-over revert-interval number |
    forced-timeout |
    id id_num |
    ldap
      {
      cn name_str |
      dn name_str |
      port port_num |
      server-name { ip_addr | name_str }
      } |
    radius
      {
      attribute acct-session-id length number |
      compatibility rfc-2138 |
      port port_num |
      retries number |
      secret shar_secret |
      timeout number  |
      zone-verification
      } |
    securid
      {
      auth-port port_num |
      duress number |
      encr id_num |
      retries number |
      timeout number
      } |
    server-name { ip_addr | name_str } |
    src-interface interface |
    tacacs { port port_num  | secret shar_secret }
    timeout number |
    type { ldap | radius | securid | tacacs } |
    username
      {
      domain dom_name |
      separator string number number
      }
    }
```

Keywords and Variables

Variables

set auth-server *name_str* […]

name_str Identifies the object name of the authentication server.

Example: The following command creates a server object name (radius1) and specifies type RADIUS:

set auth-server radius1 type radius

account-type

set auth-server *name_str* account-type { [admin] **|** [802.1X] [auth] [l2tp] [xauth] }
unset auth-server *name_str* account-type { [admin] **|** [802.1X] [auth] [l2tp] [xauth] }

account-type Specifies the types of users authenticated by the server (*name_str*).

- 802.1X specifies that the server configuration uses only 802.1x protocol for wireless connectivity between the device and the authentication server.
- admin specifies admin users.
- auth specifies authentication users.
- l2tp specifies Layer 2 Tunneling Protocol (L2TP) users.
- xauth specifies XAuth users.

You can define a user as a single user type—an admin user, an authentication user, an L2TP user, or an XAuth user. You can combine auth, L2TP, and XAuth user types to create an auth-L2TP user, an auth-XAuth user, an L2TP-XAuth user, or an auth-L2TP-XAuth user. You cannot combine an admin user with another user type.

Specify admin users only for TACACS+ authentication.

all

get auth-server all

all Specifies all configured authentication servers.

backup1 | backup2

set auth-server *name_str* { backup1 { *ip_addr* **|** *name_str* } **|** backup2 { *ip_addr* **|** *name_str* } }
unset auth-server *name_str* { backup1 **|** backup2 }

backup1 The IP address or DNS name of the primary backup authentication server for an LDAP, RADIUS, SecurID, or TACACS+ server type.

backup2 The IP address or DNS name of the secondary backup authentication server for an LDAP, RADIUS, or TACACS+ server type. SecurID does not support more than one backup server.

Example: With the following commands, you first create a RADIUS authentication server object named “radius1” at IPv6 address 2ffe::2. This server stores authentication user accounts. Then you define a primary backup server at IPv4 address 10.1.1.51 and a secondary backup server at IPv4 address 10.1.1.52.

```
set auth-server radius1 server-name 2ffe::2
set auth-server radius1 type radius
set auth-server radius1 account-type auth
set auth-server radius1 backup1 10.1.1.51
set auth-server radius1 backup2 10.1.1.52
```

fail-over

set auth-server *name_str* fail-over revert-interval *number* |
unset auth-server *name_str* fail-over revert-interval

fail-over	This feature specifies the interval (expressed in seconds) that must pass after an authentication attempt, before the device attempts authentication through backup authentication servers. When an authentication request sent to a primary server fails, the security device tries the backup servers. If authentication via a backup server is successful, and the revert-interval time interval has elapsed, the device sends subsequent authentication requests to the backup server. Otherwise, it resumes sending the requests to the primary server. The range is 0 seconds (disabled) to 86400 seconds. This feature applies to RADIUS, LDAP, and TACACS+ servers only.

forced-timeout

set auth-server forced-timeout *number*|
unset auth-server forced-timeout

forced-timeout	Specifies the time, in minutes, after which access for the authenticated user is terminated. The auth table entry for the user is removed, as are all associated sessions for the auth table entry. Forced timeout behavior is independent of idle timeout setting. The default is 0 (disabled), the range is 0 to 10000 (6.9 days). Compare “timeout” on page 54.

id

get auth-server id *id_num*
set auth-server *name_str* id *id_num*
unset auth-server id *id_num*

id	The user-defined identification number (*id_num*) of the authentication server. If you do not define an ID number, the security device creates one automatically.

Example: The following command creates an identification number (200) for the authentication server radius1:

```
set auth-server radius1 id 200
```

ldap

```
set auth-server name_str ldap
    { cn name_str | dn name_str | port port_num | server-name name_str }
```

ldap

Configures the security device to use an LDAP server for authentication.

- cn *name_str* The Common Name identifier used by the LDAP server to identify the individual entered in a LDAP server. For example, an entry of “uid” means “user ID” and “cn” means “common name”.
- dn *name_str* The Distinguished Name identifier is the path used by the LDAP server before using the common name identifier to search for a specific entry. (For example, c = us;o = juniper, where “c” stands for “country”, and “o” for “organization”.).
- port *port_num* Specifies the port number to use for communication with the LDAP server. The default port number for LDAP is 389.
- **server-name** *name_str* The IPv4 address, IPv6 address, or DNS name of the LDAP server.

Example: For an example of these options, see “Defining an LDAP Server Object” on page 57.

radius

```
set auth-server name_str radius { ... }
unset auth-server name_str radius { port | timeout }
```

radius

Configures the security device to use a RADIUS server for authentication.

- **attribute** Specifies settings for RADIUS accounting.

Each time an XAuth user connects to the device and the device authenticates the user, the device establishes a new acct-session-id, which identifies the accounting session. The accounting session lasts between the time the device sends the RADIUS server an Accounting-Start message, and the time it sends an Accounting-Stop message. To identify the user, each RADIUS access or request message contains the calling-station-id.

  - **acct-session-id length** *number* The length of the account-session-id in bytes. The acct-session-id uniquely identifies the accounting session. The default length of this value is 11 bytes. The *number* setting is for accommodating some RADIUS servers, which may have problems with the default length. You can set the length of acct-session-id from 6 bytes to 10 bytes, inclusive. To restore the default setting, execute the following command:
    unset auth-server *name_str* **radius attribute acct-session-id** *length*

- **compatibility rfc-2138** Makes RADIUS accounting comply with RFC 2138, as compared with RFC 2865. For operations where RFC 2865 (the most recent standard) and RFC 2138 are mutually exclusive, the command works in accordance with RFC 2138, instead of RFC 2865. In cases where the behavior is additive, the command works compatibly with both RFC 2865 and RFC 2138.
- port *port_num* The port number on a RADIUS server to which the security device sends authentication requests. The default port number is 1645. You can change the default port number to any number between 1024 and 65535, inclusive.
- **retries** *number* The number of retries sent to the RADIUS server before RADIUS authentication fails. The range is 1 to 20 retries.

- secret *shar_secret* Specifies the RADIUS shared secret (*shar_secret*) that is shared between the security device and the RADIUS server. The security device uses this secret to encrypt the user's password that it sends to the RADIUS server.
- timeout *number* The interval (in seconds) that the security device waits before sending another authentication request to the RADIUS server if the previous request does not elicit a response. The default is three seconds.

Example: For an example of these options, see "Defining a RADIUS Server Object" on page 56.

securid

set auth-server *name_str* securid { auth-port *port_num* **|** duress *number* **|** encr *id_num* **|** retries *number* **|** timeout *number* }

securid

Configures the security device to use a SecurID server for authentication.

- auth-port *port_num* Specifies the port number to use for communications with the SecurID server. The default SecurID port number is 5500.
- duress { 0 | 1 } If the SecurID server is licensed to use Duress mode, a value of 0 deactivates it and 1 activates it. When Duress mode is activated, a user can enter a special duress PIN number when logging in. The security device allows the login, but sends a signal to the SecurID server, indicating that someone is forcing the user to login against his or her will. The SecurID auth server blocks further login attempts by that user until he or she contacts the SecurID server admin.
- encr { 0 | 1 } Specifies the encryption algorithm for SecurID network traffic. A value of 0 specifies SDI, and 1 specifies DES. We recommend the default encryption type DES.
- retries *number* Specifies the number of retries between requests for authentication.
- timeout *number* Specifies the length of time (in seconds) that the security device waits between authentication retry attempts.

Example: For an example of these options, see "Defining a SecurID Server Object" on page 56.

server-name

set auth-server *name_str* server-name { *ip_addr* **|** *name_str* }

server-name

The IPv4 address, IPv6 address, or DNS name of the authentication server.

src-interface

set auth-server *name_str* src_interface *interface*

src-interface

Instructs the device to transmit authentication requests (RADIUS or SecurID) through the specified interface.

timeout

```
set auth-server name_str timeout number
unset auth-server name_str timeout
```

timeout	Specifies how many minutes must elapse after termination of an authentication, L2TP, or XAuth user's last session before the user needs to reauthenticate. The default timeout value is 10 minutes, and the maximum setting is 255 minutes. If the user initiates a new session before the countdown reaches the timeout threshold, he does not have to reauthenticate himself and the timeout countdown resets. If the user is an admin user, this setting specifies how many minutes of inactivity must elapse before the security device times out and closes an admin session. The default is 10 minutes and the maximum is 1000 minutes.

Example: For an example of this option, see “Defining a SecurID Server Object” on page 56.

tacacs

```
set auth-server name_str tacacs { ... }
unset auth-server name_str tacacs { port | secret }
```

tacacs — Configures the security device to use a TACACS+ server for authentication.

- **port** *port_num* The TCP port on a TACACS+ server to which the security device sends authentication requests. The default port number is 49. You can change the default port number to any number between 1024 and 32767, inclusive.
- **secret** *shar_secret* Specifies the TACACS+ shared secret (*shar_secret*) that is shared between the security device and the TACACS+ server. The security device uses this secret to encrypt the user's password that it sends to the TACACS+ server.

timeout

```
set auth-server name_str timeout number
unset auth-server name_str timeout
```

timeout	Specifies how many minutes must elapse after the termination of an authentication, L2TP, or XAuth user's last session before the user needs to reauthenticate. The default timeout value is 10 minutes, and the maximum setting is 255 minutes. If the user initiates a new session before the countdown reaches the timeout threshold, the user does not have to reauthenticate and the timeout countdown resets. If the user is an admin user, this setting specifies how many minutes of inactivity must elapse before the security device times out and closes an admin session. The default is 10 minutes and the maximum is 1000 minutes. Compare with “forced-timeout” on page 51.

Example: For an example of this option, see “Defining a SecurID Server Object” on page 56.

type

set auth-server *name_str* type { ldap **|** radius **|** securid }

type

Specifies the type of authentication server—LDAP, SecurID or RADIUS. The unset command sets type to radius.

Example: For an example of this option, see “Defining a RADIUS Server Object” on page 56.

username

set auth-server *name_str* username domain *dom_name*

set auth-server *name_str* username separator string number *number*

unset auth-server *name_str* username domain

unset auth-server *name_str* username separator

username

Specifies a domain name for a particular auth server, or a portion of a username from which to strip characters. If you specify a domain name for the auth server, it must be present in the username during authentication.

The device uses a separator character to identify where stripping occurs. Stripping removes all characters to the right of each instance of the specified character, plus the character itself. The device starts with the right most separator character.

The parameters for this feature are as follows:

- *string* is the character separator.
- *number* is the number of character separator instances with which to perform the character stripping.

If the specified number of separator characters (*number*) exceeds the actual number of separator characters in the username, the command stops stripping at the last available separator character.

Note: The device performs domain-name matching before stripping.

Example: In the following example, you strip characters to the right of two instances of a separator character in a username.

- Auth server name *Acme_Server*
- Username *bob@hello@jnpr.com*
- Separator is @
- Number of instances *2*

set auth-server Acme_Server username separator bob@hello@jnpr.com number 2

The resulting username is bob.

zone-verification

set auth-server *name_str* radius zone-verification
unset auth-server *name_str* radius zone-verification

zone-verification Verifies the zones the user is a member of and the zone configured on the port.

An authentication check can include support for zone verification. This command requires the specified RADIUS server to support RADIUS VSA enhancement. Authentication is allowed only if the zone configured on the port is a zone that a user is a member of.

In your dictionary file, add an attribute name of Zone_Verification as a string attribute type. The vendor ID is 3224, and the attribute number is 10.

Example: For an example of this option, see “Defining a RADIUS Server Object” on page 56.

Defining a RADIUS Server Object

The following commands define an auth-server object for a RADIUS server:

```
set auth-server radius1 type radius
set auth-server radius1 account-type auth l2tp xauth
set auth-server radius1 server-name 10.1.1.50
set auth-server radius1 backup1 10.1.1.51
set auth-server radius1 backup2 10.1.1.52
set auth-server radius1 radius port 4500
set auth-server radius1 radius timeout 4
set auth-server radius1 radius secret A56htYY97kl
set auth-server radius1 radius zone-verification
save
```

If you are using vendor-specific attributes, you must load the netscreen.dct file on the RADIUS server.

Defining a SecurID Server Object

The following commands define an auth-server object for a RADIUS server:

```
set auth-server securid1 type securid
set auth-server securid1 server-name 10.1.1.100
set auth-server securid1 backup1 10.1.1.110
set auth-server securid1 timeout 60
set auth-server securid1 account-type admin
set auth-server securid1 securid retries 3
set auth-server securid1 securid timeout 10
set auth-server securid1 securid auth-port 15000
set auth-server securid1 securid encr 1
set auth-server securid1 securid duress 0
save
```

Defining an LDAP Server Object

The following commands define an auth-server object for an LDAP server:

```
set auth-server ldap1 type ldap
set auth-server ldap1 account-type auth
set auth-server ldap1 server-name 10.1.1.150
set auth-server ldap1 backup1 10.1.1.151
set auth-server ldap1 backup2 10.1.1.152
set auth-server ldap1 timeout 40
set auth-server ldap1 ldap port 15000
set auth-server ldap1 ldap cn cn
set auth-server ldap1 ldap dn c=us;o=juniper;ou=marketing
save
```

The following command lists all auth-server settings:

```
get auth-server all
```

Defining a TACACS+ Server Object

The following commands define an auth-server object for a TACACS+ server:

```
set auth-server tacacs1 type tacacs
set auth-server tacacs1 account-type admin
set auth-server tacacs1 server-name 10.1.1.50
set auth-server tacacs1 backup1 10.1.1.51
set auth-server tacacs1 backup2 10.1.1.52
set auth-server tacacs1 tacacs port 1050
set auth-server tacacs1 tacacs secret A56htYY97kl
set auth-server tacacs1 timeout 4
save
```

chassis

Use the set chassis commands to activate the audible alarm feature or to set the normal and severe temperature thresholds for triggering temperature alarms.

Syntax

get

```
get chassis
```

set

```
set chassis
    {
    audible-alarm
      { all | battery | fan-failed | power-failed | temperature } |
    temperature-threshold
      { alarm | severe }
        { celsius number | fahrenheit number }
```

Arguments

audible-alarm

Enables or disables the audible alarm to announce hardware failure events.

- **all** Enables or disables the audible alarm in the event of a fan failure, an interface module failure, a power supply failure, or a temperature increase above an admin-defined threshold.
- **battery** Enables or disables the audible alarm in the event of battery failure.
- **fan-failed** Enables or disables the audible alarm in the event of a fan failure.
- **module-failed** Enables or disables the audible alarm in the event of an interface module failure.
- **power-failed** Enables or disables the audible alarm in the event of a power supply failure.
- **temperature** Enables or disables the audible alarm if the temperature rises above an admin-defined threshold.

temperature-threshold

Defines the temperature (**celsius** or **fahrenheit**) required to trigger a regular alarm or a severe alarm. A severe alarm sounds a greater frequency of audible alarms and generates a greater number of event log entries.

common-criteria

Use the **common-criteria** commands to disable all internal commands. Only the root admin can set this command. If someone other than the root admin tries to set this command, the security device displays an error message.

Syntax

set

```
set common-criteria no-internal-commands
```

Keywords and Variables

no-internal-commands

```
set common-criteria no-internal-commands
unset common-criteria no-internal-commands
```

no internal commands

Disables all internal commands.

config

Use the **config** commands to display the configuration settings for a security device or interface.

You can display recent configuration settings (stored in RAM), or saved configurations (stored in flash memory).

Syntax

exec

```
exec config { lock { abort | end | start } | rollback [ enable | disable ] }
```

get

```
get config [ all | datafile | lock | rollback | saved ]
```

set

```
set config lock timeout number
```

Keywords and Variables

all

```
get config all
```

all	Displays all configuration information.

datafile

```
get config datafile
```

datafile	Displays the NetScreen-Security Manager (NSM) datafile, which resides on the security device and contains current device configurations formatted according to the NSM syntax schema. ScreenOS generates the datafile from the current device configuration when the NSM management system queries the device.

lock

```
exec config lock start
exec config lock end
exec config lock abort
set config lock timeout number
unset config lock timeout
```

lock

Instructs the security device to lock a configuration file in memory for a specified time interval.

- **exec config lock** Locks/unlocks the configuration file in memory. You can also abort the lockout and immediately restart the device with the configuration file that was previously locked in memory.
- **set config lock timeout** Changes the default lockout period, which is 5 minutes.

rollback

```
exec config rollback
exec config rollback enable
exec config rollback disable
get config rollback
```

rollback

- **exec config rollback** Reverts the security device to the LKG (last-known-good) configuration—providing that an LKG configuration is available.
  - **enable** Enables the security device to automatically rollback to the LKG configuration in case of a problem when loading a new configuration.
  - **disable** Disables the automation of the configuration rollback feature on the security device. If you disable the automation of this feature, you can still perform a configuration rollback manually using the exec config rollback command.
- **get config rollback** Indicates if an LKG configuration is available for configuration rollback and also indicates if the automatic config rollback feature is enabled.

  If there is an LKG configuration saved in memory, the output of the command displays:
  “$lkg$.cfg” (the name of the LKG file).

  The config rollback feature is enabled if the output of the command displays “ = yes” at the end of the string. For example:
  ""$lkg$.cfg"" = yes”

  If the feature is not enabled, the output displays a blank space instead of “yes”.

saved

```
get config saved
```

counter

Use the **counter** commands to clear or display the values contained in traffic counters.

Traffic counters provide processing information, which you can use to monitor traffic flow. security devices maintain the following categories of counters:

- Screen counters, for monitoring firewall behavior for the entire zone or for a particular interface
- Policy counters, for reporting the amount of traffic affected by specified policies
- Hardware counters, for monitoring hardware performance and tracking the number of packets containing errors
- Flow counters, for monitoring the number of packets inspected at the flow level

Syntax

clear

```
clear [ cluster ] counter
      {
      all |
      ha |
      screen [ interface interface | zone zone ]
      }
```

get

```
get counter
      {
      flow | statistics
         [ interface interface | zone zone ] |
      screen { interface interface | zone zone }
      policy pol_num { day | hour | minute | month | second }
      }
```

Keywords and Variables

cluster

clear [cluster] counter [...]

cluster — Propagates the clear operation to all other devices in an NSRP cluster.

Example: To clear the contents of all counters and propagate the operation to all devices in the cluster:

clear cluster counter all

flow

get counter flow [...]

flow — Specifies counters for packets inspected at the flow level. A flow-level inspection examines various aspects of a packet to gauge its nature and intent.

ha

clear [cluster] counter ha

ha — Specifies counters for packets transmitted across a high availability (HA) link between two security devices. An HA-level inspection keeps count of the number of packets and packet errors.

interface

clear [cluster] counter screen interface *interface*

interface The name of the interface. Specifies counters for packets inspected at the interface level. The inspection checks for packet errors and monitors the quantity of packets according to established threshold settings. For more information, see "Interfaces" on page 319.

policy

get counter policy *pol_num* { day **|** hour **|** minute **|** month **|** second }

policy Identifies a particular policy (*pol_num*). This allows you to monitor the amount of traffic that the policy permits.

day | **hour** | **minute** | **month** | **second** Specifies the period of time for monitoring traffic permitted by a particular policy.

screen

clear [cluster] counter screen [interface *interface* **|** zone *zone*]
get counter screen { interface *interface* **|** zone *zone* }

screen Clears the screen counters. The **interface** *interface* parameter specifies the name of a particular interface. For more information, see "Interfaces" on page 319.

statistics

get counter statistics [...]

statistics Displays the counter statistics.

zone

get counter screen zone *zone*

zone Identifies the zone, and specifies counters for packets inspected at the zone level. The inspection checks for packet errors and monitors the quantity of packets according to established threshold settings. For more information, see "Interfaces" on page 319.

cpu-limit

Use the **cpu-limit** commands to enable and configure the CPU limit feature, which allows you to configure a more fair distribution of CPU resources.

Before you configure CPU limit feature parameters, use must use the **set cpu-limit** command to initialize and allocate resources for the feature.

Use the **get cpu-limit** command to review CPU limit feature parameters configured with the **set cpu-limit** commands.

Syntax

exec

```
exec cpu-limit mode { fair | shared }
```

get

```
get cpu-limit [ utilization ]
```

set

```
set cpu-limit
    [
    enable |
    fair-to-shared
      {
      automatic [ threshold number ] [ hold-down-time number ] |
      fair-time number |
      never
      } |
    shared-to-fair threshold number [ hold-down-time number ]
    ]
```

Keywords and Variables

enable

```
set cpu-limit enable
unset cpu-limit enable
```

enable	Use this command after configuring the CPU limit feature parameters to enable the feature.

Example: The following command enables the CPU limit feature:

set cpu-limit enable

fair-to-shared

set cpu-limit fair-to-shared automatic [threshold *number*] [hold-down-time *number*]
set cpu-limit fair-to-shared fair-time *number*
set cpu-limit fair-to-shared never
unset cpu-limit fair-to-shared { ... }

fair-to-shared	Configures parameters to determine when the security device transitions from Fair to Shared mode. ■ **automatic**: Specifies that the security device automatically transitions to Shared mode when the flow CPU utilization percentage falls below a specific threshold. ■ Optionally, specify the threshold value, which is from 0 through 100 percent. If you do not specify a threshold value, the threshold is the same value as the shared-to-fair threshold. ■ Optionally, specify a hold-down time, which is the minimum amount of time that the flow CPU utilization percentage is below the flow CPU utilization percentage threshold. Valid value range is 0 through 1800 seconds (30 minutes). The default value is 20 seconds. ■ **fair-time**: Specifies the amount of time the security device is in Fair mode before going back to Shared mode. The value range is 5 through 7200 seconds (2 hours). The default value is 30 seconds. ■ **never**: Specifies that the security device never transitions from Fair to Shared mode. You can manually force the security device into Shared mode by using the **exec cpu-limit mode shared** command.

The following command configures the security device to remain in Fair mode for 3600 seconds (1 hour).

set cpu-limit fair-to-shared fair-time 3600

mode

exec cpu-limit mode { fair **|** shared }

fair	Forces the security device into Fair mode.
shared	Forces the security device into Shared mode.

Depending on network conditions and the configured CPU limit feature parameters, the security device might transition from the mode specified by this command. Use the **exec cpu-limit mode shared** command to return to Shared mode in the following situations:

- You configured the security device to never transition from Fair to Shared mode.
- You want the security device to return to Shared mode before the specified fair-time value or hold-down time elapses.

If you configured a hold-down time with the **set cpu-limit shared-to-fair** command, use the **exec cpu-limit mode fair** command if you want the security device to return to Fair mode before the hold-down time elapses.

The following command forces the security device into Fair mode:

exec cpu-limit mode fair

The following command forces the security device into Shared mode:

exec cpu-limit mode shared

shared-to-fair threshold

```
set cpu-limit shared-to-fair threshold number [ hold-down-time number ]
unset cpu-limit shared-to-fair threshold
```

shared-to-fair threshold	Configures the flow CPU utilization percentage threshold at which the security device transitions from shared mode to Fair mode. The value range is 0 through 100. The default value is 80 percent. Optionally, configure a hold-down time, which is the minimum amount of time that the flow CPU utilization percentage must exceed the flow CPU utilization percentage threshold. Valid value range is 0 through 1800 seconds (30 minutes). The default value is 5 seconds.

The following command configures that the security device transitions from Shared to Fair mode when the flow utilization percentage stays above 70 percent for longer than 30 seconds:

set cpu-limit shared-to-fair threshold 70 hold-down-time 30

utilization

```
get cpu-limit utilization
```

utilization	Displays flow CPU utilization for the last 60 seconds. Entries with an asterisk indicate that the security device was in Fair mode.

The following command displays the flow CPU utilization for the last 60 seconds:

get cpu-limit utilization

delete

Use the delete commands to delete persistent information in flash memory.

Syntax

```
delete [ cluster ]
    {
    crypto { auth-key | file } |
    file dev_name:filename |
    node_secret ipaddr ip_addr |
    nsmgmt keys |
    pki object-id number |
    ssh device all
    }
```

Keywords and Variables

crypto

delete [cluster] crypto auth-key
delete [cluster] crypto file

crypto	Removes encrypted items from flash memory. ■ **auth-key** Removes image signature verification key. ■ **file** Remove all crypto hidden files.

file

delete file *dev_name:filename*

dev_name:filename	Deletes the file residing on the module named *dev_name* from the flash card memory.
filename	Identifies the file name.

Example: The following command deletes a file named **myconfig** in the flash memory on the memory board:

clear file flash:myconfig

node_secret ipaddr

delete node_secret ipaddr *ip_addr*

node_secret ipaddr

Deletes the SecurID stored node secret. The node secret is a 16-byte key shared between the SecurID Ace server and its clients (which may include the security device). The server and the clients use this key to encrypt exchanged traffic. The Ace Server sends the node secret to the security device during initial authentication.

The node secret *must* remain consistent with the ACE Server. Otherwise, there can be no communication between the security device and the ACE Server. You can detect communication problems by checking the ACE Server log for a message saying that the node secret is invalid. If you find such a message, the solution is as follows.

- Execute **delete node_secret**.
- On the ACE Server, change the configuration for the client (the security device) to say that the server did *not* send the node secret.

This causes the security device to request the node secret, and authorizes the ACE Server to send a new one. This action resyncs communication.

The **ipaddr** *ip_addr* parameter clears the node secret associated with the outgoing IP address of the interface that communicates with the SecurID server (*ip_addr*).

nsmgmt

delete nsmgmt keys

nsmgmt keys	Deletes the public and private keys for nsmgmt. The security device uses these keys to encrypt and decrypt the Configlet file.

pki object-id

delete pki object-id *id_num*

pki obect-id	Deletes a particular PKI object (*id_num*).

ssh device all

delete ssh device all

ssh device all — Clears all sessions and keys and disables SSH for all vsys on the device. The information removed includes:

- Active SSH sessions
- SSH enablement for the current vsys
- PKA keys
- Host keys

dhcp

Use the dhcp command to release an IP address (or all IP addresses) in the DHCP address pool.

Syntax

clear dhcp { server **|** client }

Arguments

server **|** client	Clears all IP addresses in the DHCP server or client IP address pool.

dip

Use the **dip** commands to set up a Dynamic IP (DIP) group, display DIP group information, or assign the same IP address from a port-translating DIP pool to a host that originates multiple concurrent sessions (sticky dip).

A DIP group contains one or more DIP pools, each pool consisting of a range of IP addresses defined on a Layer 3 security zone interface, Layer 3 security zone extended interface, or numbered tunnel interface. When multiple security devices are in a high availability cluster, a policy requiring source address translation and referencing a DIP pool defined on one virtual security interface (VSI) can result in dropped traffic. When that traffic arrives at a physical security device on which the DIP pool specified in the policy belongs to a VSI in an inactive virtual security device (VSD), the device drops the traffic because it cannot find the specified DIP pool to use for address translation. If, instead, the policy references a DIP group that contains DIP pools on different egress VSIs, the security device receiving the traffic can use the DIP pool belonging to the VSI for its active VSD.

NOTE: If the range of addresses in a DIP pool is in the same subnet as the interface IP address, the pool must exclude the interface IP address, router IP addresses, and any mapped IP or virtual IP addresses (MIPs and VIPs) that might also be in that subnet. If the range of addresses is in the subnet of an extended interface, the pool must exclude the extended interface IP address.

Syntax

get

```
get dip [ all ]
```

set

```
set dip
    {
    alarm-raise number1 [ alarm-clear number2 ] |
    group { id_num1 [ member id_num2 ] } |
    sticky
    }
```

Keywords and Variables

alarm-raise

set dip alarm-raise *number1* [alarm-clear *number2*]
unset alarm-raise

alarm-raise	Sets a DIP utilization alarm threshold, expressed as a percentage of possible DIP utilization. When DIP utilitzation exceeds this threshold, the device triggers an SNMP trap. Because this threshold is zero by default, it is not enabled until you increase the setting to a nonzero value. (Possible values are 50 to 100, inclusive). The alarm-clear setting specifies an optional threshold, also expressed as a percentage of possible DIP utilitzation. When DIP utilization falls below this threshold, (and DIP utilitzation previously exceeded the alarm-raise threshold), the device triggers an SNMP alarm. The default value for this threshold is 10 percent below the configured alarm-raise threshold. (Possible configured values are 40 to 100, inclusive.) The device logs these alarm events.

Example: The following command specifies upper and lower DIP utilization alarm thresholds. The device generates an SNMP alarm when either of the following conditions apply:

- DIP utilization exceeds 85 percent of capacity.
- DIP utilization falls below 45 percent of capacity.

```
set dip alarm-raise 85 alarm-clear 45
```

group

set dip group *id_num1* [member *id_num2*]
unset dip group *id_num1* [member *id_num2*]

group	Creates a DIP group or adds a DIP pool to a group. *id_num1* is the identification number you assign to the new DIP group. member *id_num2* specifies the identification number of a DIP pool.

Example: The following commands create DIP pools and a DIP group.

- DIP pool with ID 5 for interface ethernet3, which has IP address 1.1.1.1/24
- DIP pool with ID 6 for interface ethernet3:1, which has IP address 1.1.1.2/24
- DIP group with ID number 7. Both DIP pools added to the DIP group

```
set interface ethernet3 dip 5 1.1.1.10 1.1.1.10
set interface ethernet3:1 dip 6 1.1.1.11 1.1.1.11
set dip group 7
set dip group 7 member 5
set dip group 7 member 6
```

sticky

```
set dip sticky
unset dip sticky
```

sticky	Specifies that the security device assigns the same IP address to a host for multiple concurrent sessions.

dns

Use the **dns** commands to configure Domain Name System (DNS) or to display DNS configuration information.

DNS allows network devices to identify each other using domain names instead of IP addresses. Support for DNS is provided by a DNS server, which keeps a table of domain names with associated IP addresses. For example, using DNS makes it possible to reference locations by domain name (such as www.juniper.net) in addition to using the routable IP address.

DNS translation is supported in all the following applications:

- Address Book
- Syslog
- Email
- WebTrends
- Websense
- LDAP
- SecurID
- RADIUS
- TACACS +
- NetScreen-Global PRO

Before you can use DNS for domain name/address resolution, you must enter the addresses for DNS servers (the primary and secondary DNS servers) in the security device.

Syntax

clear

```
clear [ cluster ] dns
```

exec

```
exec dns refresh
```

get

```
get dns
    {
    host { cache | report | server-list | settings } |
    name dom_name
    }
```

set

```
set dns host
    {
    dns1 ip_addr |
    dns2 ip_addr |
    schedule time [ interval number ]
    }
```

Keywords and Variables

cluster

```
clear [ cluster ] dns
```

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

host

```
get dns host { ... }
set dns host { ... }
unset dns host { ... }
```

host

- **cache** Displays the DNS cache table.
- **dns1** *ip_addr* Specifies the IPv4 or IPv6 address of the primary DNS server.
- **dns2** *ip_addr* Specifies the IPv4 or IPv6 address of the bavkup DNS server.
- **name** The domain name of the host, listed in the DNS table.
  - Using the **name** option with **get** directs the security device to look up an IP address for the given domain name.
  - Using the **name** option with set places an entry in the DNS table, representing a host device with a hostname and an IP address. This allows you to reach the host from the security device using the hostname. For example, executing **set dns host name acme 3ffe::1:2:3ff:fe04** creates a DNS table entry for a host at IPv6 address **3ffe::1:2:3ff:fe04**, with a hostname of **acme**. This allows you to reach the host from the security device, as with the command **ping acme**.

Note: The DNS table is local to the security device and functions only as a proxy for the actual DNS server. Consequently, other network nodes cannot query the listed names using the security device. The main purpose of the table is to let you create an alias for an external host and to access that host from the security device.

- **report** Displays the DNS lookup table.
- **schedule** *time* Specifies the time of day to refresh DNS entries. The format of this parameter is hh:mm. The interval number parameter specifies a 4-, 6-, 8-, or 12-hour interval between DNS table refresh operations. The default interval is 24 hours; that is, once a day at the scheduled DNS lookup time. Use this option to refresh the DNS table more frequently.
- **settings** Displays DNS settings, including IP addresses, refresh setting, and the number of UDP sessions.

Example 1: The following command sets up a host as the primary DNS server at IPv6 address 3ffe::a:b:cff:fed1:

```
set dns host dns1 3ffe::a:b:cff:fed1
```

Example 2: The following command schedules a refresh time at 23:59 each day:

```
set dns host schedule 23:59
```

proxy

```
get dns proxy
set dns proxy [ enable ]
unset dns proxy [ enable ]
```

proxy	Initializes or deletes the DNS proxy. Initialization allocates all resources needed for the proxy. The **enable** switch enables or disables the DNS proxy itself. The DNS proxy feature provides a transparent mechanism that allows clients to make split DNS queries. The proxy redirects the DNS queries selectively to specific DNS servers, according to partial or complete domain specifications. This is useful when VPN tunnels or PPPoE virtual links provide multiple network connectivity, and it is necessary to direct some DNS queries to one network, and other queries to another network. The most important advantages of a DNS proxy are as follows. ■ Domain lookups are usually more efficient. For example, DNS queries meant for the corporate domain (such as **marketing.acme.com**) could go to the corporate DNS server, while all others go to the ISP DNS server, thus reducing the load on the corporate server. ■ DNS proxy can prevent domain information from leaking into the internet, thus preventing malicious users from learning about internal network configuration.

refresh

```
exec dns refresh
```

refresh	Refreshes all DNS entries. Using the option directs the security device to perform a manual DNS lookup.

server-select

```
clear [ cluster ] dns server-select domain dom_name
get dns server-select
set dns server-select domain dom_name [ outgoing-interface interface { ... } ]
```

server-select

Identifies external DNS servers according to all or part of the fully qualified domain name (FQDN) contained in each DNS query. This process is called *proxy DNS*.

- **primary-server** *ip_addr* specifies the IPv4 or IPv6 address of the primary proxy DNS server.
- **secondary-server** *ip_addr* specifies the IPv4 or IPv6 address of the secondary proxy DNS server.
- **tertiary-server** *ip_addr* specifies the IPv4 or IPv6 address of the tertiary proxy DNS server.

The **failover** switch directs the DNS to fail over to another server if the currently active server fails.

Use the **set dns server-select** commands to create a partial or full entry for a DNS proxy domain lookup. Such entries allow the security device to selectively direct DNS queries to different DNS servers. For example, you can direct all DNS queries with FQDNs containing a particular domain name to a corporate server, and direct all other DNS queries to an ISP server. To denote these other, unspecified queries, use the asterisk symbol (see example below).

The optional **outgoing-interface** parameter specifies the interface through which the security device transmits the DNS query.

Note: You can make such queries secure by specifying a tunnel interface.

For more information about proxy DNS, see “proxy” on page 73.

Example: The following commands create two proxy-DNS entries that selectively forward DNS queries to different servers.

- All DNS queries for FQDNs containing the domain name acme.com go through interface tunnel.1, to the DNS server at IPv6 address 3aff:7483:ad33::33ff:fe34. For example, the DNS proxy could query this server for the FQDN intranet.acme.com.
- All other DNS queries go out through interface ethernet3 to the DNS server at IPv4 address 1.1.1.23.

```
set dns proxy
set dns proxy enable
set dns server-select domain .acme.com outgoing-interface tunnel.1 primary-server
    3aff:7483:ad33::33ff:fe34
set dns server-select domain * outgoing-interface ethernet3 primary-server 1.1.1.23
```

domain

Use the domain commands to set or display the domain name of the security device.

A *domain name* is a character string that identifies the NetSceen device. This name allows other devices to access the security device through a DNS server, thus identifying the device without using an explicit IP address.

Syntax

get

get domain

set

set domain *name_str*

Keywords and Variables

Variables

name_str	Defines the domain name of the security device.

Example: The following command sets the domain of the security device to acme:

set domain acme

dot1x

Use the dot1x commands to review 802.1X session information and clear 802.1X sessions. You can also clear 802.1X statistics.

Use the get dot1x command to review 802.1X configured parameters for all interfaces.

Syntax

clear

clear dot1x { session [id *number*] **|** statistics }

get

get dot1x [session [id *number*] **|** statistics]

Keywords and Variables

session

clear dot1x session [id *number*]
get dot1x [session [id *number*]]

session	Specifies all 802.1X sessions or detailed information about a specific 802.1X session. Use the **get dot1x session** command to see a list of session IDs. Use a session ID and the optional **id** keyword to see details for a particular session or to clear it.

Example: The following command clears the 802.1X session with an ID of 54:

clear dot1x session id 54

statistics

clear dot1x statistics
get dot1x statistics

statistics	Displays all 802.1X-enabled interface statistics or clears all 802.1X statistics.

Example: The following command clears all 802.1X statistics:

clear dot1x statistics

Syntax

exec downgrade

Keywords and Variables

None.

Downgrades and NetScreen-Security Manager

envar

Use the **envar** commands to define environment variables.

The security device uses environment variables to make special configurations at startup.

Syntax

get

get envar [resource]

set

set envar *string*

Keywords and Variables

Variables

set envar *string*
unset envar *string*

string	The location of the environment variables files.

Example 1: The following command defines the location of the system configuration as file2.cfg in slot2:

set envar config=slot2:file2.cfg

Example 2: To enable IPv6 on the security device, enter the following command:

set envar ipv6=yes

resource

get envar resource

resource	Displays the following information: ■ (max-session) Maximum number of sessions ■ (max-sa) Maximum number of security associations (SAs) ■ (max-l2tp-tunnel) Maximum number of L2TP tunnels

event

Use the **event** commands to display or clear event log messages.

The *event log* monitors and records system events and network traffic. The security device categorizes logged system events by the following severity levels:

- **Alert**: Messages for multiple user authentication failures and other firewall attacks not included in the emergency category.
- **Critical**: Messages for URL blocks, traffic alarms, high availability (HA) status changes, and global communications.
- **Debugging**: All messages.
- **Emergency**: Messages concerning SYN attacks, Tear Drop attacks, and Ping of Death attacks.
- **Error**: Messages for admin log on failures.
- **Information**: Any kind of message not specified in other categories.
- **Notification**: Messages concerning link status changes, traffic logs, and configuration changes.
- **Warning**: Messages for admin logins and logouts; failures to log in and log out; and user authentication failures, successes, and timeouts.

The event log displays the date, time, level, and description of each system event.

Syntax

clear

```
clear [ cluster ] event [ end-time time ]
```

get

```
get event
    [ module name_str ]
      [ level
        {
        alert |
        critical |
        debug |
        emergency |
        error |
        information |
        notification |
        warning
        }
      ]
        [ type id_num1 ] [ -id_num2 ]
          [ start-date date [ time ] ] [ end-date date [ time ] ]
            [ start-time time ] [ end-time time ]
              [ include string ] [ exclude string ]
```

```
                [src-ip ip_addr1 [ -ip_addr2 | src_netmask mask ] ]
                  [dst-ip ip_addr1 [ -ip_addr2 | dst_netmask mask ] ]sort-by
      {
      date
        [ start-date date [ time ] ]
          [ end-date date [ time ] ]
      dst-ip [ ip_addr [ -ip_addr | dst-netmask mask ] ]
      src-ip [ ip_addr [ -ip_addr | src-netmask mask ] ]
      time
        [ start-time time ]
          [ end-time time ]
      }
]
```

Keywords and Variables

cluster

clear cluster event [...]

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

dst-ip

get event dst-ip *ip_addr* [*-ip_addr* | dst-netmask *mask*]
get event sort by dst-ip [*ip_addr* [*-ip_addr* | dst-netmask *mask*]
]

dst-ip	Directs the device to display event logs with the specified destination IP address or address range. The device can also sort event logs by destination IP address.

include | exclude

get event [...] [include *string*] [exclude *string*] [...]

include exclude	Directs the device to exclude or include events containing a specifies string of characters (*string*).

level

get event module *name_str* level { ... }

level

Specifies the priority level of the event message. The priority levels are as follows:

- **emergency** (Level 0) The system is unusable.
- **alert** (Level 1) Immediate action is necessary.
- **critical** (Level 2) The event affects functionality.
- **error** (Level 3) Error condition exists.
- **warning** (Level 4) The event might affect functionality.
- **notification** (Level 5) The event is a normal occurrence.
- **information** (Level 6) The event generates general information about normal operation.
- **debug** (Level 7) The event generates detailed information for troubleshooting purposes.

module

get event module name_str [...]

module

Specifies the name of the system module that generated the event.

src-ip

get event src-ip *ip_addr1* [*-ip_addr2* | src-netmask *mask*]
get event sort by src-ip *ip_addr1* [*-ip_addr2* | src-netmask *mask*]

src-ip

Directs the device to sort event logs by source IP address. The device can also display event logs with the specified source IP address or address range.

start-time | end-time

clear [cluster] event end-time *time*
get event [...] [start-time *time*] [end-time *time*] [...]

end-time *time*
start-time *time*

Specifies the lower and upper ends of a range of times for an event. When you specify a start-time and/or end-time, the device sorts or filters the event logs based on the specified times, regardless of the date. The format is: *hh:mm:ss*.

When you use the **end-time** option with the **clear event** command, you specify the date and optionally the time in the following format: *mm/dd/yy-hh:mm:ss*.

Example: The following command clears all events generated before May 1, 2002 at 11:30am:

get event end-time 05/01/02-11:30:00

start-date | end-date

get event [start-date *date* [*time*]] [end-date *date* [*time*]]
get event sort-by *date* [start-date *date* [*time*]] [end-date *date* [*time*]]

start-date
end-date

Specifies the lower and upper ends of a range of times for an event. The format is:

mm/dd/yy-hh:mm:ss

You can omit the year (the current year is the default), or express the year using the last two digits or all four digits. The hour, minute, and second are optional. The delimiter between the date and the time can be a dash or an underscore:

12/31/2001-23:59:00

12/31/2001_23:59:00

type

get event module *name_str* level { ... } type *id_num1* [...]

type

Specifies a priority level or a range of priority levels.

Chapter 2

exit Through nsmgmt

This section lists and describes the ScreenOS command line interface (CLI) commands **exit** through **nsmgmt**.

NOTE: Certain commands and features are platform-dependent and might be unavailable on your Juniper Networks security product platform. Check your product datasheet for feature availability.

Click on a command to view details:

exit
failover
file
flow
gate
group
group-expression
hostname
ike
ike-cookie
interface
ip
ip-classification
ippool
l2tp
license-key
log
mac
mac-learn
memory
mirror
ndp
nsmgmt

exit

Use the **exit** command to exit a command context or a virtual system or to terminate and log out from a CLI session.

Syntax

```
exit
```

Keywords and Variables

None.

Example: The following **exit** command exits the context of policy ID 1 and returns the command context to the top command level:

```
device-> set policy id 1
device(policy:1)-> set dst-addr 2.2.2.5/32
device(policy:1)-> exit
device->
```

NOTE: When issuing the exit command at the top command level (not from within a command context), you must log back into the console to configure a security device.

failover

Use the **failover** commands to configure failover settings on the security device.

Syntax

get

```
get failover
```

set

```
set failover
      {
      auto |
       enable |
      holddown number [ recover number ] |
      type { route vrouter vrouter ip_addr/mask | track-ip | tunnel-if }
      }
```

exec

```
exec failover
      {
      force |
      revert
      }
```

Keywords and Variables

auto

```
set failover auto
unset failover auto
```

auto

Directs the security device to automatically fail over from the primary interface to the backup and from the backup interface to the primary. By default, failover is manual (the administrator must use the CLI or WebUI to switch from the primary interface to the backup and from the backup interface to the primary).

force

```
exec failover force
```

force

Forces traffic to be switched to the backup interface.

holddown

```
set failover holddown number
unset failover holddown
```

holddown

Specifies the time interval (*number*), in seconds, the security device delays failover actions. This value has an effect in the following situations:

- The security device switches traffic to the backup interface.
- The security device switches traffic from the backup interface to the primary interface, when the primary interface becomes available again.

The default holddown interval 30 seconds.

Example: The following command sets a failover delay of 45 seconds:

```
set failover holddown 45
```

revert

```
exec failover revert
```

revert

Forces traffic to be switched from the backup interface to the primary.

type

```
set failover type { track-ip | tunnel-if }
```

type

Specifies the type of event that determines interface failover. You can specify the following types:

- track-ip instructs ScreenOS to use IP tracking to determine failover.
- tunnel-if instructs ScreenOS to use VPN tunnel status to determine failover.

file

Use the **file** commands to clear or display information for files stored in the flash memory.

Syntax

clear

clear [cluster] file *dev_name:filename*

get

get file [*filename* **|** info]

Keywords and Variables

Variables

clear [...] file *dev_name:filename*
get file *filename*

dev_name:filename	Deletes the file with the name *filename* from the flash card memory.
filename	Defines the file name stored in the flash card memory.

Examples: The following command deletes a file named **myconfig** in the flash memory on the memory board:

clear file flash:myconfig

The following command displays information for the file named **corpnet** from the flash card memory:

get file corpnet

cluster

clear cluster file *dev_name:filename*

cluster	Propagates the **clear** operation to all other devices in an NSRP cluster.

info

get file info

info	Displays the base sector and address.

flow

Use the **flow** commands to determine how the security device manages packet flow.

The device can regulate packet flow in the following ways:

- Enable or disable DNS replies when there is no matching DNS request
- Pass or block packets containing destination MAC addresses that are not in the MAC learning table
- Set or display the initial session timeout values
- Control or prevent packet fragmentation

Syntax

get

```
get flow [ perf | tcpmss ]
```

set

```
set flow
    {
    aging { early-ageout number | high-watermark number | low-watermark number }
    all-tcp-mss [ number ] |
    allow-dns-reply |
    check tcp-rst-sequence |
    gre-in-tcp-mss |
    gre-out-tcp-mss |
    hub-n-spoke-mip |
    initial-timeout number |
    mac-cache mgt |
    mac-flooding |
    max-frag-pkt-size number |
    multicast |
    no-tcp-seq-check |
    path-mtu |
    route-change-timeout |
    syn-proxy syn-cookie |
    tcp-mss [ number ] |
    tcp-rst-invalid-session |
    tcp-syn-bit-check |
    tcp-syn-check |
    tcp-syn-check-in-tunnel |
    vpn-tcp-mss [ number ]
    }
```

Keywords and Variables

aging

set flow aging early-ageout *number*
set flow aging { high-watermark *number* **|** low-watermark *number* }
unset flow aging { early-ageout **|** high-watermark **|** low-watermark }

aging	Directs the security device to begin aggressively aging out sessions when the number of entries in the session table exceeds the high-watermark setting, and then stop when the number of sessions falls below the low-watermark setting. When the session table is in any other state, the normal session timeout value is applied—for TCP, session timeout is 30 minutes; for HTTP, it is 5 minutes; and for UDP, it is 1 minute. During the time when the aggressive aging out process is in effect, the security device ages out sessions—beginning with the oldest sessions first—at the rate you specify. ■ early-ageout *number* Defines the ageout value before the security device aggressively ages out a session from its session table. The value you enter can be from 2 to 10 units, each unit representing a 10-second interval. The default early-ageout value is 2, or 20 seconds. ■ high-watermark *number* Sets the point at which the aggressive aging out process begins. The number you enter can be from 1 to 100 and indicates a percentage of the session table capacity in 1-percent units. The default is 100 (100 percent). ■ low-watermark *number* Sets the point at which the aggressive aging out process ends. The number you enter can be from 1 to 10 and indicates a percentage of the session table capacity in 10-percent units. The default is 10 (100 percent).

Example: The following commands activate the aggressive aging-out process when the session table reaches 70 percent of capacity and deactivate the process when it drops below 60 percent, and set the aggressive ageout value at 30 seconds:

```
set flow aging low-watermark 60
set flow aging high-watermark 70
set flow aging early-ageout 3
```

allow-dns-reply

set flow allow-dns-reply
unset flow allow-dns-reply

allow-dns-reply	Allows an incoming DNS reply packet without a matched request. If allow-dns-reply is disabled and an incoming UDP first-packet has dst-port 53, the device checks the DNS message packet header to verify that the QR bit is 0 (which denotes a query message). If the QR bit is 1 (which denotes a response message) the device drops the packet, does not create a session, and increments the 'illegal pak' flow counter for the interface. By default, allow-dns-reply is disabled. Enabling allow-dns-reply directs the device to skip the check.

all-tcp-mss

set flow all-tcp-mss *number*
unset flow all-tcp-mss

all-tcp-mss	Sets the TCP-maximum segment size (TCP-MSS) value for all TCP SYN packets with an MSS value greater than that specified by the **tcp-mss** option (described below). The device resets the value whether the packets are clear-text or encrypted. By contrast, use the **tcp-mss** option to modify the MSS value only when it is necessary to avoid fragmentation caused by an IPSec operation. If you enter the **set flow tcp-mss** command, that setting overrides the **all-tcp-mss** option for VPN traffic. The **set flow tcp-mss** command is in turn overridden when **set flow vpn-tcp-mss** is used.

gre-in-tcp-mss

set flow gre-in-tcp-mss
unset flow gre-in-tcp-mss

gre-in-tcp-mss	Specifies inbound GRE TCP-MSS option (64-1420).

gre-out-tcp-mss

set flow gre-out-tcp-mss
unset flow gre-out-tcp-mss

gre-out-tcp-mss	Specifies outbound GRE TCP MSS option (64-1420).

hub-n-spoke-mip

set flow hub-n-spoke-mip
unset flow hub-n-spoke-mip

hub-n-spoke-mip	Permits the security device to use hub-and-spoke policies for MIP traffic, when the traffic loops back on the same physical interface. This option only has an effect when the interface is bound to the Untrust zone.

initial-timeout

set flow initial-timeout *number*
unset flow initial-timeout

initial-timeout	Defines the length of time in seconds (*number*, expressed in 10-second intervals) that the security device keeps an initial session in the session table before dropping it, or until the device receives a FIN or RST packet. The range of time is from 20 seconds (setting of 2) to 300 seconds (setting of 30).

Example: The following command sets the initial timeout value to 1 minute (60 seconds):

```
set flow initial-timeout 6
```

mac-flooding

```
set flow mac-flooding
unset flow mac-flooding
```

mac-flooding — Enables the security device to pass a packet across the firewall even if its destination MAC address is not in the MAC learning table.

max-frag-pkt-size

set flow max-frag-pkt-size *number*
unset flow max-frag-pkt-size

max-frag-pkt-size — The maximum allowable size for a packet fragment generated by the security device. You can set the *number* value between 1024 and 1500 inclusive.

For example, if a received packet is 1540 bytes and **max-frag-pkt-size** is 1460 bytes, the device generates two fragment packets. The first is 1460 bytes and the second is 80 bytes. If you reset **max-frag-pkt-size** to 1024, the first fragment packet is 1024 bytes and the second is 516 bytes.

Example: The following command sets the maximum size of a packet generated by the security device to 1024 bytes:

```
set flow max-frag-pkt-size 1024
```

no-tcp-seq-check

```
set flow no-tcp-seq-check
unset flow no-tcp-seq-check
```

no-tcp-seq-check — Skips the sequence number check in stateful inspection.

path-mtu

```
set flow path-mtu
unset flow path-mtu
```

path-mtu — Enables path-MTU (maximum transmission unit) discovery. If the security device receives a packet that must be fragmented, it sends an ICMP packet suggesting a smaller packet size.

perf

```
get flow perf
```

perf — Displays the perf information.

route-change-timeout

set flow route-change-timeout *number*
unset flow route-change-timeout *number*

route-change-timeout Sets and unsets the the session timeout value on a route change to a nonexistent route. You can set *number* between 6 and 1800 seconds inclusive. Unsetting this keyword removes the route-change-timeout value, causing sessions to time out based on their original timeout, if a route change occurs and no new route is found.

If route-change is not set, the current behavior is maintained, and sessions discovered to have no route are aged out using their current session timeout values.

syn-proxy syn-cookie

set flow syn-proxy syn-cookie
unset flow syn-proxy syn-cookie

syn-proxy syn-cookie Sets the flow from the traditional SYN Proxy mode to SYN Cookie mode. SYN Cookie is enabled globally on the security device and is activated when the configured syn-flood attack-threshold is exceeded.

tcp-mss

tcp-rst-invalid-session get flow tcpmss
set flow tcpmss [*number*]
unset flow tcpmss

tcp-mss Sets the TCP-maximum segment size (TCP-MSS) value for all TCP SYN packets for outbound VPN traffic only. The security device modifies the MSS value in the TCP packet to avoid fragmentation caused by the IPSec operation. The value set using **set flow vpn-tcp-mss** command takes precedence over the one set by **set flow tcp-mss**.

tcp-rst-invalid-session

set flow tcp-rst-invalid-session
unset flow tcp-rst-invalid-session

tcp-rst-invalid-session Directs the device to immediately terminate and reset a session when TCP packets have enabled reset bits.

tcp-syn-bit-check

set flow tcp-syn-bit-check
unset flow tcp-syn--bit-check

tcp-syn-bit-check Checks the TCP SYN bit before forwarding the packet to the processor for session creation. SYN bit check is done in the application-specific integrated circuit (ASIC), and the packet is forwarded to the processor for session creation. This improves throughput performance. If the SYN bit is not set in the first packet received, the security device drops the packet. This is the default.

The **tcp-syn-bit-check** feature is a subset of **tcp-syn-check**; if you want to enable just **tcp-syn-bit-check**, you must disable **tcp-syn-check**, which is disabled by default.

The Resulting Behavior column in the following table shows the effect or effects of various combinations of TCP SYN bit-screening configuration.

1. Check the TCP SYN bit before creating a session in the processor.

2. Check the TCP SYN bit before creating a session is in the ASIC.

3. Refresh the session after the three-way handshake.

4. Check the TCP SYN bit on traffic from the tunnel

TCP SYN configuration is the same on all security devices, whether ASIC-based or not.

tcp-syn-check	tcp-syn-bit-check	tcp-syn-check-in-tunnel	Resulting Behavior
enable	n/a	enable	2,3,4
enabled	n/a	disabled	1,3
disabled	enabled	enabled	2,4
disabled	enabled	disabled	1
disabled	disabled	enabled	none
disabled	disabled	disabled	none

tcp-syn-check

set flow tcp-syn-check
unset flow tcp-syn-check

tcp-syn-check Checks the TCP SYN bit before creating a session , and refreshes the session after the TCP three-way handshake. If the SYN bit is not set, the security device drops the packet.

The tcp-syn-check feature is a superset of tcp-syn-bit-check; therefore enabling tcp-syn-check enables tcp-syn-bit-check as well. If you want to enable just tcp-syn-bit-check, you must disable tcp-syn-check.

tcp-syn-check-in-tunnel

set flow tcp-syn-check-in-tunnel
unset flow tcp-syn-check-in-tunnel

tcp-syn-check-in-tunnel Checks the TCP SYN bit before creating a session for tunneled packets. By default, the security device checks that the SYN bit is set in the first packet of a VPN session. If it is not set, the security device drops it.

vpn-tcp-mss

```
set flow vpn-tcp-mss [ number ]
unset flow vpn-tcp-mss
```

vpn-tcp-mss	Sets the TCP-maximum segment size (TCP-MSS) value for all TCP SYN packets for both outbound and inbound VPN traffic. **Note:** For VPN traffic, the **tcp-mss** or **vpn-tcp-mss** command can be used. If both commands are set, the latter command has precedence over the former. If both commands are not set, **all-tcp-mss** command is used. You need to unset the **vpn-tcp-mss** command to make the device use the **tcp-mss** command. When the **vpn-tcp-mss** command is used, an MSS value configured by the **tcp-mss** command will not be valid even if the **tcp-mss** command is run again.

Defaults

The default initial timeout value is 2 (20 seconds).

The MAC-flooding feature is enabled by default.

gate

Use the **gate** command to check the number of gates on the security device, how many are in use, and how many are still available.

Gates are logical access points in the firewall for FTP and similar applications. The security device creates the gates, then converts a gate for each new session when data traffic occurs.

Syntax

```
get gate
```

Keywords and Variables

None.

Defaults

The number of default gates varies by device. For more information, refer to the datasheet for your security device.

group

Use the **group** commands to group several addresses or several services under a single name.

A *group* allows you to reference a group of addresses or services by a single name in a policy. This eliminates the need for a separate policy for each address or service. For example, you can create a service group that includes FTP, HTTP, and HTTPS services, and then reference that group in a policy.

NOTE: Although a single policy might reference a service group with three members, the security device generates multiple internal rules from that policy. Overusing address and service groups with high member counts can unexpectedly consume internal resources.

Syntax

get

```
get group { address zone [ grp_name ] | service [ grp_name ] }
```

set

```
set group
     {
     address zone grp_name [ ipv6 ] [ add name_str ] [ comment string ] |
     service grp_name [ add name_str ] [ comment string ]
     }
```

Keywords and Variables

add

```
set group address zone grp_name [ ipv6 ] [ add mbr_name ] [ comment string ]
set group service grp_name [ add mbr_name [ comment string ] ]
```

add *name_str*	Adds an address or service named *mbr_name*.

Examples: The following command creates an address group named **engineering** for the Trust zone and adds the address **hw-eng** to the group:

set group address trust engineering add hw-eng

The following command creates a service group named **inside-sales** and adds the service AOL to the group:

set group service inside-sales add AOL

address

```
get group address zone [ … ]
set group address zone grp_name [ … ]
unset group address zone grp_name [ … ]
```

address	Performs the operation on an address group. The *zone* value specifies the zone to which the address group is bound. This zone is either a default security zone or a user-defined zone. For more information, see “Zones” on page 321.

Example: The following command creates an empty address group (named headquarters) for the Trust zone:

set group address trust headquarters

clear

unset group address *zone grp_name* clear
unset group service *grp_name* clear

clear	Removes all the members of an address or service group.

Example: The following command removes all members from an address group (engineering) bound to the Trust zone:

unset group address trust engineering clear

comment

set group address *zone grp_name* […] [comment *string*]
set group service *grp_name* […] [comment *string*]

comment	Adds a comment *string* to the service group or address group entry.

Example: The following command creates an address group named engineering for the Trust zone, adds the address hw-eng to the group, and includes a comment about the group:

set group address trust engineering add hw-eng comment "Engineering Group"

remove

unset group address *zone grp_name* remove *name_str*
unset group service *grp_name* remove *name_str*

remove	Removes the address (or service) named *name_str*. If you do not specify an address (or service) group member, the unset group { address | service } command deletes the entire address group or service group.

Example: The following command removes the address admin-pc from the engineering address group:

unset group address trust engineering remove admin-pc

service

get group service *grp_name*
set group service *grp_name* […]
unset group service *grp_name* […]

service *grp_name*	Performs the operation on a service group.

Example: The following command creates an empty service group and names it web_browsing:

set group service web_browsing

Notes

Each address group and service group you create must have a unique name. You cannot use the same address group name as a service group name.

You cannot add the predefined address or service named "any" to a group.

While a policy references a group, you cannot remove the group, although you can modify it.

From the console, you can add only one member to a group at a time.

group-expression

Use the **group-expression** commands to set up or display group expressions for use in security policies.

A *group expression* allows you to include or exclude users or user groups, according to NOT, AND, or OR operators. Such expressions are only usable for external users and user groups.

Syntax

get

```
get group-expression
    {
    name_str |
    all |
    id number
    }
```

set

```
set group-expression name_str
    {
    not name_str |
    name_str { and | or } name_str |
    id number |
    }
```

Keywords and Variables

Variables

```
get group-expression name_str
set group-expression name_str
unset group-expression name_str
```

name_str	The name of the group expression.

all

get group-expression all

all	Specifies all group expressions.

and | or

set group-expression *name_str name_str* and *name_str*
set group-expression *name_str name_str* or *name_str*

and | or	Specifies AND or OR relationship between users, user groups, or group expressions.

Example: The following commands create group expressions SalesM and SM_Group, place them in an OR relationship, and then place SM_Group and Office_1 in an AND relationship:

```
set user-group Sales_Group location external
set user-group Marketing_Group location external
set group-expression SalesM Sales_Group or Marketing_Group
set group-expression SM_Group Office_1 and SalesM
```

id

get group-expression id *number*
set group-expression *name_str* id *number*
unset group-expression id *number*

id *number*	Specifies an identification number for the group expression.

not

set group-expression *name_str* not *name_str*

not	Specifies negation.

Example: The following command creates a NOT group expression that does not allow the Office_1 user:

```
set group-expression Total_Users not Office_1
```

hostname

Use the **hostname** commands to define the security device name. This name always appears in the console command prompt.

The hostname is a character string that identifies the security device. If you define a hostname for the device (such as ns500gate) and a domain name for the device (such as “netscreen,” using the **domain** command), you can use the hostname and domain name (ns500gate.netscreen) as a gateway for a VPN tunnel.

Syntax

get

```
get hostname
```

set

```
set hostname string
```

Keywords and Variables

Variables

string	Sets the name of the security device.

Example: The following command changes the security device hostname to acme:

set hostname acme

ike

Use the **ike** commands to define the Phase 1 and Phase 2 proposals and the gateway for an AutoKey IKE (Internet Key Exchange) VPN tunnel, and to specify other IKE parameters.

To establish an AutoKey IKE IPSec tunnel between peer devices, two phases of negotiation are required:

- In Phase 1, the peer devices establish a secure channel in which to negotiate the IPSec SAs.

- In Phase 2, the peer devices negotiate the IPSec SAs for encrypting and authenticating the ensuing exchanges of user data.

The gateway definition identifies the devices or remote users with which the security device establishes the VPN tunnel.

Syntax

exec

```
exec ike preshare-gen name_str usr_str
```

get

```
get ike
    {
    accept-all-proposal |
    ca-and-type |
    cert |
    conn-entry |
    cookies |
    gateway [ name_str ] |
    heartbeat |
    id-mode |
    initial-contact [ all-peers | single-gateway [ name_str ] ] |
    initiator-set-commit |
    member-sa-hold-time |
    p1-max-dialgrp-sessions |
    p1-proposal name_str |
    p1-sec-level |
    p2-proposal name_str |
    p2-sec-level |
    policy-checking |
    respond-bad-spi |
    responder-set-commit |
    soft-lifetime-buffer
    }
```

set

Phase 1 Proposal

```
set ike p1-proposal name_str
    [ dsa-sig | rsa-sig | preshare ]
      [ group1 | group2 | group5 ]
        { esp
          { 3des | des | aes128 | aes192 | aes256
            { md5 | sha-1
              [
              days number |
              hours number |
              minutes number |
              seconds number
              ]
            }
          }
        }
```

Phase 2 Proposal

```
set ike p2-proposal name_str
    [ group1 | group2 | group5 | no-pfs ]
      {
      esp [ 3des | des | aes128 | aes196 | aes256 | null ] |
      ah
      }
        [ md5 | null | sha-1
          [
          days number |
          hours number |
          minutes number |
          seconds number ]
          ]
            [ kbyte number ]
        ]
    }
```

Gateway Tunnel

```
set ike gateway name_str
    {
    address { ip_addr | hostname[.dom_name ] [ id ] }
    dialup { usr_str | grp_name } |
    dpd
      {
      always-send |
      interval number1 |
      retry number2
      } |
    dynamic
      {
      string |
      asn1-dn { [ container string ] [ wildcard string ] |
      fqdn string |
      ip-addr string |
      u-fqdn string
      } |
    }
        [ aggressive | main ] [ local-id id_str ]
          [ outgoing-interface interface
            [ outgoing-zone zone ]
          ]
              [ preshare key_str | seed-preshare key_str ]
                {
                sec-level { basic | compatible | standard } |
                proposal name_str1
                  [ name_str2 ] [ name_str3 ] [ name_str4 ]
                } |
    modecfg
      {
      client
        {
        action update-dhcpserver |
        admin-preference number |
        pd interface interface  [ sla-id id_num [ sla-len number ] ] |
      server { action add-route | info-origin local dns }
```

IKE Heartbeat

```
set ike gateway name_str heartbeat
    {
    hello number |
    threshold number |
    reconnect number
    }
```

Certificates

```
set ike gateway name_str cert
    {
    my-cert id_num |
    peer-ca [ id_num | all ] |
    peer-cert-type { pkcs7 | x509-sig }
    }
```

NAT-Traversal

```
set ike gateway name_str nat-traversal
    [
    udp-checksum |
    keepalive-frequency number
    ]
```

XAuth

```
set ike gateway name_str xauth
    [
    client { any | chap | securid } username name_str password name_str |
    server name_str
       [ chap ] [ query-config ] [ user name_str | user-group name_str ] |
    bypass-auth
    ]
```

Other IKE Command Switches

```
set ike
    {
    accept-all-proposal |
    id-mode { ip | subnet } |
    initial-contact
       [
       all-peers |
       single-gateway name_str
       ] |
    initiator-set-commit |
    member-sa-hold-time number |
    p1-max-dialgrp-sessions { count number | percentage number } |
    policy-checking |
    respond-bad-spi spi_num |
    responder-set-commit |
    single-ike-tunnel name_str |
    soft-lifetime-buffer number
    }
```

Keywords and Variables

accept-all-proposal

```
get ike accept-all-proposal
set ike accept-all-proposal
unset ike accept-all-proposal
```

accept-all-proposal Directs the security device to accept all incoming proposals. By default, the device accepts only those proposals matching predefined or user-defined proposals. This command is primarily useful when troubleshooting AutoKey IKE tunnels.

address

set ike gateway address { *ip_addr* **|** *name_str* } { ... }

address	Defines the remote IKE gateway address as an IP address, as a hostname, or as a fully qualified domain name (FQDN, which is a hostname + domain name). Use this option to set up a site-to-site VPN. Note: If you specify a hostname or an FQDN that the security device cannot resolve to an IP address, the IKE gateway is classified as disabled.

Example: The following command specifies www.juniper.net as the address of a remote IKE gateway named jn1, defines the preshared key as 7a850wq, and specifies the Phase 1 security level as compatible:

```
set ike gateway jn1 address www.juniper.net preshare 7a850wq sec-level compatible
```

NOTE: The *compatible* security level for Phase 1 negotiations includes the following four proposals: pre-g2-3des-sha, pre-g2-3des-md5, pre-g2-des-sha, and pre-g2-des-md5.

aggressive | main

set ike gateway *name_str* { ... } aggressive [...]
set ike gateway *name_str* { ... } main [...]

aggressive | main	Defines the mode used for Phase 1 negotiations. Use Aggressive mode only when you need to initiate an IKE key exchange without ID protection, as when a peer unit has a dynamically assigned IP address. Main mode is the recommended key-exchange method because it conceals the identities of the parties during the key exchange.

ca-and-type

get ike ca-and-type

ca-and-type	Displays the supported certificate authorities (CAs) and certificate types.

cert

get ike cert
set ike gateway *name_str* cert my-cert *id_num*
set ike gateway *name_str* cert peer-ca [*id_num* **|** all]
set ike gateway *name_str* cert peer-cert-type { pkcs7 **|** 509-sig

cert

Uses a digital certificate to authenticate the VPN initiator and recipient.

gateway *name_str* cert

Specifies which certificates to use.

- my-cert *name_str* Specifies a particular certificate when the local security device has multiple loaded certificates.
- peer-ca *name_str* Specifies a preferred certificate authority (CA).
- peer-cert-type { pkcs7 | x509 } Specifies a preferred type of certificate (PKCS7 or X509).

  If you set the peer-ca and peer-cert-type values, the device inserts them in any certificate request it sends to the peer. If the peer has multiple local certificates, these values help the peer select a certificate.

Note: The security device does not use the peer-ca or peer-cert-type settings to check certificates received from the peer.

If possible, the peer should send a certificate issued by the peer-ca CA. However, if the peer sends a certificate issued by a different CA, the security device searches local memory for the certificate of the issuing CA; if the search is successful, the device accepts the peer certificate. If the search is unsuccessful, the device uses a certificate issued by a different CA.

conn-entry

get ike conn-entry

conn-entry

Displays the Connection Entry Table.

cookies

get ike cookies

cookies

Displays the cookie table, and the total number of dead and active cookies.

dialup

set ike gateway *name_str* dialup { *usr_str* **|** *grp_name* } […]

dialup

Identifies an IKE dialup user (*usr_str*) or dialup group (*grp_name*). Use this option to set up a dialup VPN. To specify a user's attributes, use the set user command. (To specify dialup group attributes, use the set user-group command.)

dpd

```
get ike gateway name_str dpd
set ike gateway name_str dpd { always-send | interval number1 | retry number2 }
unset ike gateway name-str dpd [ ... ]
```

dpd

Configures the device to use DPD (Dead-Peer Detection). DPD is a protocol used by security devices to verify the current existence and availability of IPSec peer devices. A device performs this verification by sending encrypted IKE Phase 1 notification payloads (R-U-THERE) to peers, and waiting for DPD acknowledgements (R-U-THERE-ACK).

- **always-send** Instructs the device to send DPD requests regardless of whether there is outgoing IPSec traffic to the peer.
- **interval** *number1* Specifies the DPD interval. This interval is the amount of time (expressed in seconds) the device allows to pass before considering a peer to be dead. The device considers the peer dead when all of the following conditions apply after the DPD interval expires:
  - The device received no matching R-U-THERE-ACK response after sending the configured number of transmitted R-U-THERE requests to the peer.
  - There was no incoming IPSec traffic from the peer on any of the IPSec SAs.
  - The device received no R-U-THERE request from DPD peer.
- **retry** *number2* The maximum number of times to send the R-U-THERE request before considering the peer to be dead.

dynamic

set ike gateway *name_str* dynamic { … } […]

dynamic	Specifies the identifier for the remote gateway with a dynamic IP address. Use this option to set up a VPN with a gateway that has an unspecified IP address. ■ *string* A string you can use as a peer ID. ■ asn1-dn [container] [wildcard] *string* The ASN1 domain name. The container switch treats *string* as a container. The wildcard switch treats *string* as a wild card. ■ fqdn The fully qualified domain name (such as www.acme.com). ■ ip_addr *string* The IP address of the remote gateway interface. ■ u-fqdn *string* The user fully qualified domain name (such as admin@acme.com).

gateway

get ike gateway
set ike gateway *name_str* { … } […]
unset ike gateway { … }

gateway	Configures or displays settings for a remote tunnel gateway.

heartbeat

get ike heartbeat
set ike gateway *name_str* heartbeat { … }
unset ike gateway heartbeat { … }

heartbeat	Specifies the IKE heartbeat protocol parameters. ■ hello *number* Sets the IKE heartbeat protocol interval (in seconds). ■ reconnect *number* Sets the quiet interval (in seconds) that elapses before the security device reconnects a failed tunnel. ■ threshold *number* Sets the number of retries before the security device considers the connection lost and removes all Phase 1 and Phase 2 keys related to this gateway.

id-mode

get ike id-mode
set ike id-mode ip
set ike id-mode subnet

id-mode	Defines the IKE ID mode in the Phase 2 exchange as either a host (IP) address or a gateway (subnet). If you use the ip switch, the device sends no Phase 2 ID. If you choose the subnet switch, the device sends proxy Phase 2 IDs. (Use the ip switch when setting up a VPN tunnel between a security device and a CheckPoint 4.0 device. Otherwise, use the subnet switch.)

initial-contact

get ike initial-contact
set ike initial-contact [all-peers **|** single-gateway *name_str*]
unset ike initial-contact

initial-contact	Determines how the security device performs initial contact with an IKE peer. ■ Specifying **all-peers** instructs the security device to delete all SAs, then send an initial contact notification to each IKE peer. ■ Specifying **single-gateway** *name_str* instructs the security device to delete all SAs associated with the specified IKE gateway, then send an initial contact notification. If you specify none of the above options, the security device sends an initial contact notification to all peers during the first IKE single-user session after a system reset.

initiator-set-commit

get ike initiator-set-commit
set ike initiator-set-commit
unset ike initiator-set-commit

initiator-set-commit	When the security device performs as an IKE initiator, sets the commit bit in the ISAKMP header. The party who sends the last message in the exchange does not use the new IPSec SA until it receives confirmation from the other party.

local-id

set ike gateway *name_str* { … } local-id *id_str* […] { … }

local-id	Defines the IKE ScreenOS identity of the local device. The device sends this ID to the remote gateway during IKE negotiation. To instruct the security device to derive the IKE identity from the Distinguished Name (DN) in the local certificate, specify the following for **local-id** (including square brackets): [DistinguishedName] If there is more than one certificate on your security device, you may need to specify which certificate to use. For more information, see “cert” on page 102.

member-sa-hold-time

get ike member-sa-hold-time
set ike member-sa-hold-time *number*
unset ike member-hold-sa

member-sa-hold-time	The length of time (in minutes) the device keeps an unused SA allocated for a dialup user.

modecfg client

set ike gateway *name_str* modecfg client action update-dhcpserver
set ike gateway *name_str* modecfg client admin-preference *number*
set ike gateway *name_str* modecfg client pd interface interface
    [sla-id *id_num* [sla-len *number*]]
unset ike gateway *name*-str modecfg

modecfg client ModeCFG allows ISPs to delegate IPv6 prefixes or addresses to the XAuth client.

- **action update-dhcpserver** Instructs the security device to distribute the received XAuth configuration information (DNS and WINS) to DHCP servers on other interfaces and to the device itself.
- **admin-preference** *number* Sets the local preference (*number*) for configuration information learned through sources that use XAuth, PPPoE, DHCPv6, DHCP, or another protocol. The preference number determines which source to use first.

  The device can learn the DNS server addresses statically (from the WebUI or the CLI), or it can learn them dynamically from PPPoE, DHCP or XAuth. The device stores these learned addresses in the DNS server list. It then selects the best two addresses from this list and designates them as the primary and secondary DNS server addresses. The **admin-preference** *number* setting specifies how much preference the device gives to addresses learned through one source or protocol, in comparison with another source or protocol. To do this, it uses an election protocol.

  First, the device compares the **admin-preference** values. If the values differ, it selects the address with the highest value. If the values are identical, it uses the highest protocol. (The protocol levels, from highest to lowest, are PPPoE, XAuth, DHCP, and CLI, respectively.) If the protocols are identical, it chooses the address with the greatest numerical value.

- **pd** Enables the client to receive delegated prefixes and use them to configure networks or subnets on separate interfaces.
  - The **sla-id** *id_num1* parameter identifies the SLA (Site-Local Aggregate) number, an integer value that denotes a subnet of the network connected to the interface. This value can be up to 16 bits in length.
  - The **sla-len** *id_num2* parameter specifies the length of the SLA (expressed in bytes). This value can range from 0 to 16.

  If you specify a **sla-id** *id_num1* and **sla-len** *id_num2* of zero, the device automatically right-aligns the SLA to 64 bits by padding it with zeroes. For example, if the XAuth server delegates the prefix 15::/48, and the SLA is 0, the resulting prefix is 15:0:0:0::/64 (or 15::/64). If the SLA is 3, the resulting prefix is 15:0:0:3::/64.

modecfg server

set ike gateway name_str modecfg server { action add-route **|** info-origin local dns }
unset ike gateway name_str modecfg server action add-route
    { action add-route **|** info-origin local dns }

modecfg server Allows the XAuth/Modecfg server to add a route for each IPv4 address and netmask or for each IPv6 address and prefix length assigned to the client from an upstream router.

nat-traversal

set ike gateway *name_str* nat-traversal udp-checksum
set ike gateway *name_str* nat-traversal keepalive-frequency *number*
unset ike gateway *name_str* nat-traversal [...]

nat-traversal

Enables or disables IPsec NAT Traversal, a feature that allows transmission of encrypted traffic through a security device configured for NAT. The NAT Traversal feature encapsulates ESP packets into UDP packets. This prevents the NAT device from altering ESP packet headers in transit, thus preventing authentication failure on the peer security device.

- udp-checksum enables the NAT-Traversal UDP checksum operation (used for UDP packet authentication).
- keepalive-frequency specifies the frequency (in seconds) with which the security device sends NAT-traversal keepalive messages.

Examples: The following command enables NAT traversal for a gateway named mktg:

```
set ike gateway mktg nat-traversal
```

The following command sets the Keepalive setting to 25 seconds:

```
set ike gateway mktg nat-traversal keepalive-frequency 25
```

outgoing-interface

set ike gateway *name_str* { ... } outgoing-interface *interface* [...]

outgoing-interface

Defines the interface through which the security device sends IKE traffic for this gateway.

Example: The following commands, executed on a local security device, configure the device to request service from a HTTP server in Paris. The security device exchanges all HTTP packets through a VPN tunnel.

- Remote gateway: Paris_Gateway
- Remote gateway interface IPv6 address: 3a55::130:6:aff:fe3a
- Outgoing interface: ethernet3
- Local IPv6 address for ethernet3: 21e4::e443:a1ff:feb6
- VPN tunnel (Paris_Tunnel) uses the defined gateway Paris_Gateway

```
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode host
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 21e4::e443:a1ff:feb6/64
set ike gateway Paris_Gateway address 3a55::130:6:aff:fe3a outgoing-interface
    ethernet3 local-address 21e4::e443:a1ff:feb6 proposal dsa-g2-aes128-md5
set vpn Paris_Tunnel gateway Paris_Gateway proposal g2-esp-aes128-sha
set address trust Our_Local 21e4::e443:a1ff:feb6/64
set address untrust Their_Remote 3a55::130:6:aff:fe3a/64
set policy from trust to untrust Our_Local Their_Remote http tunnel vpn
    Paris_Tunnel
```

p1-max-dialgrp-sessions

get ike p1-max-dialgrp-sessions
set ike p1-max-dialgrp-sessions count *number*
set ike p1-max-dialgrp-sessions percentage *number*
unset ike p1-max-dialgrp-sessions

p1-max-dialgrp-sessions	Displays the allowed concurrent Phase 1 negotiations for dialup groups.

p1-proposal

get ike p1-proposal *name_str*
set ike p1-proposal *name_str* […] { … }
unset ike p1-proposal *name_str*

p1-proposal

Names the IKE Phase 1 proposal, which contains parameters for creating and exchanging session keys and establishing Phase 1 security associations.

- dsa-sig | rsa-sig | preshare Specifies the method to authenticate the source of IKE messages. preshare refers to a preshared key, which is a key for encryption and decryption that both participants have before beginning tunnel negotiations. rsa-sig and dsa-sig refer to two kinds of digital signatures, which are certificates that confirm the identity of the certificate holder. (The default method is preshare.)
- group1 | group2 | group5 Identifies the Diffie-Hellman group, a technique that allows two parties to negotiate encryption keys over an insecure medium; such as, the Internet. Group2 is the default group.
- esp Specifies Encapsulating Security Payload protocol, which provides encryption and authentication.
- des | 3des | aes128 | aes192 | aes256 Specifies the encryption algorithm.
- md5 | sha-1 Specifies the authentication (hashing) algorithm used in ESP protocol. The default algorithm is SHA-1, the stronger of the two algorithms.
- The following parameters define the elapsed time between each attempt to renegotiate a Phase 1 security association. The minimum allowable lifetime is 180 seconds. The default lifetime is 28800 seconds.
  - days *number*
  - hours *number*
  - minutes *number*
  - seconds *number*

Example: The following command defines a Phase 1 proposal named sf1.

- Preshared key and a group 1 Diffie-Hellman exchange
- Encapsulating Security Payload (ESP) protocol using the 3DES and MD5 algorithms
- Lifetime of 3 minutes

```
set ike p1-proposal sf1 preshare group1 esp 3des md5 minutes 3
```

p1-sec-level

```
get ike p1-sec-level
```

p1-sec-level	Displays the predefined IKE Phase 1 proposals in descending order of security level.

p2-sec-level

```
get ike p2-sec-level
```

p2-sec-level	Displays the predefined IKE Phase 2 proposals in descending order of security level.

p2-proposal

```
get ike p2-proposal name_str
set ike p2-proposal name_str [ ... ] { ... }
set ike p2-proposal name_str
```

p2-proposal

Names the IKE Phase 2 proposal. This proposal defines parameters for creating and exchanging a session key to establish a security association (SA).

- **group1 | group2 | group5 | no-pfs** Defines how the security device generates the encryption key. Perfect Forward Secrecy (PFS) is a method for generating each new encryption key independently from the previous key. Selecting **no-pfs** turns this feature off, so IKE generates the Phase 2 key from the key generated in the Phase 1 exchange. If you specify one of the Diffie-Hellman groups, IKE automatically uses PFS when generating the encryption key. The default is Group 2.
- **ah | esp** In a Phase 2 proposal, identifies the IPSec protocol.
  - **esp [des | 3des | aes128 | aes192 | aes256]** Specifies Encapsulating Security Payload (ESP) protocol, which provides both encryption and authentication. Specifies the encryption algorithm used in ESP protocol. (The default protocol is **des**.)
  - **ah** Specifies Authentication Header (AH) protocol, which provides authentication only.
- **md5 | null | sha-1** Specifies the authentication (hashing) algorithm used in ESP or AH protocol. The default algorithm is MD5 for non-FIPS mode, and SHA is the default for FIPS mode. The null switch specifies no authentication.
- The following parameters define the elapsed time between each attempt to renegotiate a security association. The minimum allowable lifetime is 180 seconds. The default lifetime is 28800 seconds.
  - **days** *number*
  - **hours** *number*
  - **minutes** *number*
  - **seconds** *number*
- **kbytes** *number* Indicates the maximum allowable data flow in kilobytes before ScreenOS renegotiates another security association. The default value is **0** (infinity).

Example: The following command specifies Phase 2 proposal g2-esp-3des-null.

- Group 2 Diffie-Hellman exchange
- ESP using 3DES without authentication
- Lifetime of 15 minutes

set ike p2-proposal g2-esp-3des-null group2 esp 3des null minutes 15

policy-checking

get ike policy-checking
set ike policy-checking
unset ike policy-checking

policy-checking	Checks to see if the policies of the two peers match before establishing a connection. Use policy checking when configuration on the peer gateways support multiple tunnels. Otherwise, the IKE session fails. You can disable policy checking when only one policy is configured between two peers.

preshare

set ike p1-proposal *name_str* preshare […]

preshare	Directs the device to use preshared key authentication for IKE Phase 1 negotiation. In this mode, both peer devices use a shared password to generate a encryption and decryption key.

set ike gateway *name_str* { ... } [...] preshare *key_str*

preshare	Specifies the Preshared key (*key_str*) used in the Phase 1 proposal. (If you use an RSA- or DSA-signature in the Phase 1 proposal, do not use this option).

Example: For an example of this option, see "Setting Up a Policy-Based VPN Tunnel" on page 116.

preshare-gen

exec ike preshare-gen *name_str usr_str*

preshare-gen	Generates an individual preshared key for a remote dialup user associated with a Group IKE ID user. The security device generates each preshared key from a seed value (specified in the command set ike gateway). After the device generates the preshared key, you can use it to set up a configuration for the remote user. (Remove any spaces.) ■ *name_str* is the IKE gateway name. To create such a gateway, use the set ike gateway *name_str* command. ■ *usr_str* is the full IKE ID of an individual user, which belongs to a Group IKE ID user. To create such a user, use the set user *name_str* ike-id command. The Group IKE ID user must be associated with a dialup user group to support a group of users.

Example: The following commands create a single group IKE ID user and assign the user to a dialup user group. Then they create VPNs and policies that allow dialup users with matching partial IKE ID values to establish secure communication through the security device.

- The name of the group IKE ID user is User1, with partial IKE identity of acme.com.
- The number of dialup users that can share this user's IKE identity is 10.
- The dialup user group is Office_1.
- The seed value for creating the preshared key is jk930k.
- The Phase 1 IKE gateway defined for the server side is Corp_GW.
- The Phase 2 VPN defined for the server side is Corp_VPN.
- The Phase 1 IKE gateway defined for the client side is Office_GW.
- The Phase 2 VPN defined for the client side is Office_VPN.
- The individual user's full IKE identity is chris@acme.com.
- The trusted server that dialup users access from the outside is a webserver with IP address 1.1.110.200.

```
set user User1 ike-id u-fqdn acme.com share-limit 10
set user-group Office_1 user User1
set ike gateway Corp_GW dialup Office_1 aggressive seed-preshare jk930k
    proposal pre-g2-3des-md5
set vpn Corp_VPN gateway Corp_GW tunnel proposal g2-esp-3des-md5
set address trust http_server 1.1.110.200 255.255.255.255
set policy incoming “dial-up vpn” http_server any tunnel vpn Corp_VPN
```

To generate the preshared key for chris@acme.com:

```
exec ike preshare-gen Corp_GW chris@acme.com
```

NOTE: For this example, assume that this command generates c5d7f7c1806567bc57d3d30d7bf9b93baa2adcc6.

On the client side:

```
set ike gateway Office_GW address 10.1.10.10 aggressive local-id chris@acme.com
    preshare c5d7f7c1806567bc57d3d30d7bf9b93baa2adcc6 proposal
    pre-g2-3des-md5
set vpn Office_VPN gateway Office_GW tunnel proposal g2-esp-3des-md5
set address untrust http_server 1.1.110.200
set policy outgoing “inside any” http_server any tunnel vpn Office_VPN
```

proposal

```
set ike gateway name_str { ... } [ ... ] proposal name_str1
    [ name_str2 ] [ name_str3 ] [ name_str4 ]
```

proposal	Specifies the name (*name_str*) of a proposal. You can specify up to four Phase 1 proposals.

Example: For an example of this option, see “Setting Up a Policy-Based VPN Tunnel” on page 116.

respond-bad-spi

```
get ike respond-bad-spi
set ike respond-bad-spi [ number ]
unset ike respond-bad-spi
```

respond-bad-spi	Responds to packets with bad security parameter index (SPI) values. The specified *number* value is the number of times to respond to bad SPIs per gateway.

responder-set-commit

```
get ike responder-set-commit
set ike responder-set-commit
unset ike responder-set-commit
```

responder-set-commit	Directs the security device to set the commit bit in the ISAKMP header when the device acts as an IKE responder. The peer that sends the last message in the exchange does not use the new IPSec SA until it receives information from the other peer.

sec-level

```
set ike gateway name_str { ... } [ ... ] sec-level { ... }
```

sec-level	Specifies which predefined security proposal to use for IKE. The basic proposal provides basic-level security settings. The compatible proposal provides the most widely used settings. The standard proposal provides settings recommended by Juniper Networks.

Example: The following command specifies the predefined security proposal compatible:

```
set vpn Corp_VPN gateway Corp_GW sec-level compatible
```

seed-preshare

```
set ike gateway name_str { ... } [ ... ] seed-preshare key_str
```

seed-preshare	Specifies a seed value (*key_str*) for a user group with Preshared Key configurations. Such a configuration performs IKE authentication for multiple dialup users, each with an individual preshared key, without having a separate configuration for each user. Instead, use the seed to generate the preshared key with the exec ike preshare-gen command.

Example: The following commands configure IKE authentication for multiple dialup users in a user group:

- Interface ethernet1 bound to the Trust zone and interface ethernet3 bound to the Untrust zone
- Dialup user named User2, placed in a user group named office_2
- Gateway configuration for office_2, with a preshared key seed value of jk930k
- Security policy for all dialup users with the partial IKE identity specified for User2

```
set interface ethernet1 zone trust
set interface ethernet1 ip 10.1.1.1/24
set interface ethernet3 zone untrust
set interface ethernet3 ip 1.1.1.1/24
set address trust web1 10.1.1.5/32
set user User2 ike-id u-fqdn netscreen.com share-limit 10
set user-group office_2 user User2
set ike gateway Corp_GW dialup office_2 aggressive seed-preshare jk930k
    sec-level compatible
set vpn Corp_VPN gateway Corp_GW sec-level compatible
set policy top from untrust to trust “Dial-Up VPN” web1 http tunnel vpn Corp_VPN
save
```

single-ike-tunnel

set ike single-ike-tunnel *name_str*
unset ike single-ike-tunnel *name_str*

single-ike-tunnel	Specifies a single Phase 2 SA for all policies to a particular remote peer gateway.

Example: The following command specifies a Phase 2 SA for all policies to the peer gateway gw1:

```
set ike single-ike-tunnel gw1
```

soft-lifetime-buffer

get ike soft-lifetime-buffer
set ike soft-lifetime-buffer *number*

soft-lifetime-buffer	Sets a time interval (in seconds) before the current IPSec SA key lifetime expires. When this interval is reached, the device initiates the rekeying operation.

xauth

set ike gateway *name_str* xauth
unset ike gateway xauth

xauth	Enables XAuth authentication for the specified IKE gateway configuration.

xauth bypass-auth

set ike gateway *name_str* xauth bypass-auth

bypass-auth	Instructs the security device, acting as an XAuth server, to perform only XAuth mode-config, which assigns the XAuth client with an IP address, and DNS and WINS server settings. The XAuth client is not required to authenticate him or herself.

xauth client

set ike gateway *name_str* xauth client { any **|** chap **|** securid } username *name_str*
password *string*

client	Specifies that the security device is an XAuth client. You can specify the following authentication types: ■ **any** Instructs the device to allow any authentication type. ■ **chap** Instructs the device to allow Challenge Handshake Authentication Protocol (CHAP) only. ■ **securid** Instructs the device to allow authentication via SecurID only.
username	Specifies the username for the XAuth client to use on the XAuth server.
password	Specifies the password for the XAuth client to use on the XAuth server.

Example: The following example configures an XAuth client.

- Gateway kg1
- Any authentication type allowed
- Username kgreen and password pubs123

set ike gateway kg1 xauth client any username kgreen password pubs123

xauth server

set ike gateway *name_str* xauth server *name_str*
set ike gateway *name_str* xauth server *name_str* [chap] [query-config]
[user *name_str* **|** user-group *name_str*]
unset ike gateway xauth

server	Specifies the object name of the external server that performs the XAuth authentication. ■ **chap** Instructs the device to use Challenge Handshake Authentication Protocol (CHAP). ■ **query-config** Instructs the device to query the client configuration from the server. ■ **user** *name_str* Enables XAuth authentication for an individual user. ■ **user-group** *name_str* Enables XAuth authentication for the users in a XAuth user group.

Defaults

Main mode is the default method for Phase 1 negotiations.

The default time intervals before the ScreenOS mechanism renegotiates another security association are 28,800 seconds in a Phase 1 proposal, and 3600 seconds in a Phase 2 proposal.

The default ID mode is subnet. (Changing the ID mode to IP is only necessary if the data traffic is between two security gateways, one of which is a CheckPoint 4.0 device.)

The default soft-lifetime-buffer size is 10 seconds.

By default, the single-ike-tunnel flag is not set.

By default, the commit bit is not set when initiating or responding to a Phase 2 proposal.

Setting Up a Policy-Based VPN Tunnel

Creating a policy-based VPN tunnel for a remote gateway with a static IP address requires up to seven steps. To set up device-A at one end of a VPN tunnel for bidirectional traffic, follow the steps below:

1. Bind interfaces to zones and assign them IP addresses:

   ```
   set interface ethernet1 zone trust
   set interface ethernet1 ip 10.1.1.1/24
   set interface ethernet3 zone untrust
   set interface ethernet3 ip 1.1.1.1/24
   ```

2. Set the addresses for the end entities beyond the two ends of the VPN tunnel:

   ```
   set address trust host1 10.1.1.5/32
   set address untrust host2 10.2.2.5/32
   ```

3. Define the IKE Phase 1 proposal and Phase 2 proposal. If you use the default proposals, you do not need to define Phase 1 and Phase 2 proposals.

4. Define the remote gateway:

   ```
   set ike gateway gw1 address 2.2.2.2 main outgoing-interface ethernet3
       preshare netscreen proposal pre-g2-3des-sha
   ```

5. Define the VPN tunnel as AutoKey IKE:

   ```
   set vpn vpn1 gateway gw1 proposal g2-esp-des-md5
   ```

6. Set a default route (both the Trust and Untrust zones are in the trust-vr routing domain):

   ```
   set vrouter trust-vr route 0.0.0.0/0 interface ethernet3 gateway 1.1.1.250
   ```

7. Set outbound and inbound policies:

```
set policy from trust to untrust host1 host2 any tunnel vpn vpn1
set policy from untrust to trust host2 host1 any tunnel vpn vpn1
```

The procedure for setting up a VPN tunnel for a dialup user with IKE also constitutes up to seven steps:

1. Bind interfaces to zones and assign them IP addresses.

2. Define the protected address that you want the dialup user to be able to access through the tunnel. (See the **set address** command.)

3. Define the user as an IKE user. (See the **set user** command.)

4. Define the IKE Phase 1 proposal, Phase 2 proposal, and remote gateway. (**Note:** If you use the default proposals, you do not need to define a Phase 1 or Phase 2 proposal.)

5. Define the VPN tunnel as AutoKey IKE. (See the **set vpn** command.)

6. Set a default route (both the Trust and Untrust zones are in the trust-vr routing domain).

7. Define an incoming policy, with **Dial-Up VPN** as the source address and the VPN tunnel you configured in step 5.

ike-cookie

Use the **ike-cookie** command to remove IKE-related cookies from the security device.

Syntax

```
clear [ cluster ] ike-cookie { all | ip_addr }
```

Keywords and Variables

Variables

```
clear cluster ike-cookie ip_addr
clear ike-cookie ip_addr
```

ip_addr	Directs the security device to remove cookies based on a IP address (*ip_addr*), which can be either IPv4 or IPv6.

Example: The following command removes all cookies based on the IP address 10.1.10.10:

```
clear ike-cookie 10.1.10.10
```

all

```
clear cluster ike-cookie all
clear ike-cookie all
```

all	Directs the security device to remove all cookies.

cluster

```
clear cluster ike-cookie all
clear cluster ike-cookie ip_addr
```

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

interface

Use the interface commands to define or display interface settings for a security device.

Interfaces are physical or logical connections that handle network, virtual private network (VPN), High Availability (HA), and administrative traffic. For a description of the interfaces you can configure on a security device, see “Interfaces” on page 319.

Syntax

get

```
get interface interface
    [
    dhcp
      {
      client |
      relay |
      server { ip { allocate | idle } | option }
      } |
    dhcp6
      {
      client pd |
      server options
        {
        binding { duid string | name name_str } |
        client-duid |
        dns |
        pd |
        search-list
        } |
    dip |
    ipv6 { config | ra | inverse-nd } |
    mip |
    protocol { ospf | rip } |
    screen |
    secondary [ ip_addr ] |
    track-ip [ ip ]
    ]
```

set (Layer 3 Interfaces)

```
set interface interface
    {
    bandwidth number |
    dip id_num
        {
        ip_addr1 [ ip_addr2 ] |
        ipv6 prefix ip_addr/pref_len
        shift-from ip_addr3 [ to ip_addr4 [ ip_addr5 ] ]
        }
            [ fix-port ] |
    [ ext ip ip_addr/mask ] dip id_num
        {
        ip_addr1 |
        shift-from ip_addr2 to ip_addr3
        } [ ip_addr4 ] [ fix-port ] |
    gateway ip_addr [ no-default-interface ] |
    group |
    ip ip_addr/mask { tag id_num } |
    ipv6 inverse-nd { link-mtu }
    manage { ident-reset | nsmgmt | ping | snmp | ssh | ssl | telnet | web } |
    manage-ip ip_addr |
    mip ip_addr [ ipv6 ]   host ip_addr [ netmask mask ] [ vrouter name_str ] |
    mtu number |
    nat |
    ntp-server |
    phy
        {
        auto |
        full { 10mb | 100mb } |
        half { 10mb | 100mb } |
        holddown number |
        link-down
        } |
    route |
    route-deny |
    tag id_num zone zone |
    vip ip_addr | interface-ip ] [ + ] port_num [ name_str ip_addr [ manual ] ] |
    webauth |
    webauth-ip ip_addr |
    zone zone
    }
```

set (Layer 2 Interfaces)

```
set interface interface
    {
    manage { ident-reset | nsmgmt ping | nmp | ssh | ssl | telnet | web } |
    phy
        {
        auto |
        full { 10mb | 100mb } |
        half { 10mb | 100mb } |
        holddown number |
        link-down
        } |
    webauth
    }
```

set (DHCP Relay / Server)

```
set interface interface dhcp
    {
    relay { server-name { name_str | ip_addr } | service | vpn } |
    server
        {
        enable | auto | disable
        ip ip_addr { mac mac_addr | to ip_addr } |
        option
            {
            dns1 | dns2 | dns3 | gateway | news | nis1 | nis2 | pop3 | smtp
                { ip_addr } |
            domainname name_str |
            lease number |
            netmask mask |
            nistag name_str |
            wins1 ip_addr |
            wins2 ip_addr
            } |
        service
        }
    }
```

set (DHCP Client)

```
set interface interface dhcp client
    {
    admin-preference number |
    enable |
    settings
        {
    admin-preference number |
        autoconfig |
        lease number |
        server ip_addr |
        update-dhcpserver |
        vendor id_str
        }
    }
```

set (DHCPv6 Client)

```
set interface interface dhcp6 client
    [
    action set-default-route |
    admin-preference number |
    enable |
    options
      {
      rapid-commit |
      request { dns | pd | search-list }
      } |
    pd [ iapd-id id_num ]
      {
      prefix ip_addr/pref_len number1 [ number2 ] |
      ra-interface interface sla-id id_num1 sla-len id_num2
      } |
    prefer-server id_num
    ]
```

set (DHCPv6 Server)

```
set interface interface dhcp6 server
    {
    enable |
    options
      {
      client-duid duid string [ name name_str ] |
      dns { dns1 | dns2 | dns3 } ip_addr |
      pd
        { duid string | name string
          [ iapd-id id_num ] prefix ip_addr/pref_len number1 [ number2 ]
        }
      rapid-commit |
      search-list name name_str id_num
      }
    preference number
    }
```

set (High Availability)

```
set interface { ha | ha1 | ha2 }
    {
    bandwidth number |
    phy
      {
      auto |
      full { 10mb | 100mb } |
      half { 10mb | 100mb } |
      holddown number |
      link-down
      } |
    }
```

set (IP Tracking)

```
set interface interface track-ip
    [
    dynamic |
    ip ip_addr
        [
        interval number |
        threshold number |
        weight number
        ] |
    threshold number
    ]
```

set (IPv6)

```
set interface interface ipv6
    {
    enable |
    interface-id id_num |
    inverse-nd { link-mtu } |
    ip ip_addr/pref_len |
    mode { host | router } |
    nd
        {
        base-reachable-time number |
        dad-count number |
        nud |
        probe-time number |
        retransmit-time number
        } |
    ra
        {
    accept |
        default-life-time number |
        hop-limit number |
        link-address |
        link-mtu |
        managed |
        max-adv-int number |
        min-adv-int number |
        other |
        preference { high | low | med }
        prefix ip_addr/pref_len number1 number2 [ autonomous [ onlink ] ] |
        reachable-time |
        retransmit-time |
        transmit
        }
    }
```

set (Loopback Interface)

```
set interface interface loopback-group interface
```

set (RIP and RIPng)

```
set interface interface protocol
   {
   rip
      [
      authentication
         {
         active-md5-key-id id_num |
         md5 key_str [ key-id id_num ] |
         password pswd_str
         } |
   ripng
   }
            [
            enable |
            metric number |
            passive-mode |
            route-map name_str |
            split-horizon [ poison-reverse ]
            ]
```

set (Tunnel)

```
set interface tunnel.number
   {
   dip id_num
      {
      ip_addr1
         [ ip_addr2 ] [ fix-port ] |
      shift-from ip_addr3
      } |
   [ ext ip ip_addr/mask ] dip id_num
      {
      ip_addr1
         [ ip_addr2 ] [ fix-port ] |
      shift-from ip_addr3
      } |
   ip ip_addr/mask |
   loopback-group |
   manage-ip ip_addr |
   mip ip_addr host ip_addr
      [ netmask mask [ vrouter name_str ] ] |
   nhtb ip_addr vpn tunn_str |
   protocol { bgp | ospf } |
   route-deny |
   zone name_str
   }
```

Keywords and Variables

Variables

get interface *interface* [...]
set interface *interface* { ... } [...]

interface	The name of the interface. For more information, see "Interfaces" on page 319.

Example: The following command specifies the IP address of a remote gateway peer (1.1.1.25) for the ethernet4 interface:

set interface ethernet4 gateway 1.1.1.25

admin-preference

set interface *interface* dhcp6 client admin-preference *number*
set interface *interface* dhcp client settings admin-preference *number*
unset interface *interface* dhcp6 client admin-preference
unset interface *interface* dhcp client settings admin-preference

admin-preference

Sets the local preference (*number*) for configuration information learned through sources that use DHCP or DHCPv6. (The range is 0 to 255.)

The device can learn the DNS server addresses statically (from the WebUI or the CLI), or it can learn them dynamically (from PPPoE, DHCPv6, DHCP or XAuth). The device stores these learned addresses in the DNS server list. It then selects the best two addresses from this list, and designates them as the primary and secondary DNS server addresses. The admin-preference *number* setting specifies how much preference the device gives to addresses learned through one source or protocol, in comparison with another source or protocol. To do this, it uses an election protocol.

First, the device compares the admin-preference values. If the values differ, it selects the address with the highest value. If the values are identical, it uses the highest protocol. (The protocol levels, from highest to lowest, are PPPoE, XAuth, DHCP, and CLI respectively.) If the protocols are identical, it chooses the address with the greatest numerical value.

bandwidth

set interface *interface* bandwidth *number*
unset interface *interface* bandwidth

bandwidth	The guaranteed maximum bandwidth in kilobits per second for all traffic traversing the specified interface.

Example: The following command specifies bandwidth of 10,000 kilobits per second for interface ethernet4:

set interface ethernet4 bandwidth 10000

broadcast

set interface *interface* broadcast { flood **|** arp [trace-route] }
unset interface *interface* broadcast [arp [trace-route]]

broadcast	(vlan1 interface only.) Controls how the security device determines reachability of other devices while the device is in transparent (L2) mode. ■ flood Instructs the security device to flood frames received from an unknown host out to all interfaces that are in Transparent mode. In the process, the device might attempt to copy frames out of ports that cannot access the destination address, thus consuming network bandwidth. ■ arp [trace-route] Instructs the security device to generate an Address Resolution Protocol (ARP) broadcast. If the broadcast finds the unknown destination IP address, the device loads its ARP table with the appropriate MAC address and interface. The device uses this entry to reach the destination device directly, and only sends frames through the correct port, thus saving bandwidth. Generating the initial ARP can cause delay, but only for the first frame.

Example: The following command instructs the security device to generate an Address Resolution Protocol (ARP) broadcast:

set interface vlan1 broadcast arp

bypass-non-ip

set interface *interface* bypass-non-ip
unset interface *interface* bypass-non-ip

bypass-non-ip	(vlan1 interface only.) Allows non-IP traffic (such as IPX) to pass through a security device running in Transparent mode. (ARP is a special case for non-IP traffic. It is always passed, even if when this feature is disabled.) (bypass-non-ip-all does not allow nonbroadcast, nonmulticast traffic.)

bypass-non-ip-all

set interface *interface* bypass-non-ip-all
unset interface *interface* bypass-non-ip-all

bypass-non-ip-all	(vlan1 interface only.) Allows nonbroadcast, nonmulticast, and non-IP traffic to pass through a security device running in Transparent mode. (ARP is a special case for non-IP traffic. It is always passed, even if when this feature is disabled.)

bypass-others-ipsec

set interface *interface* bypass-others-ipsec
unset interface *interface* bypass-others-ipsec

bypass-others-ipsec	(vlan1 interface only.) Openly passes all IPSec traffic through a security device in Transparent mode. The security device does not act as a VPN tunnel gateway but passes the IPSec packets onward to other gateways.

dhcp client

```
set interface interface dhcp client enable
set interface interface dhcp client { ... }
```

dhcp client	Configures an interface for DHCP client services. ■ **enable** Enables DHCP client services for the interface. ■ **settings** Configures DHCP parameters for the interface. ■ **autoconfig** Enables automatic configuration after device power-up. ■ **lease** *number* Sets the default lease time (in minutes). ■ **server** *ip_addr* Specifies the IP address of the DHCP server. ■ **update-dhcpserver** Forwards TCP/IP settings from the DHCP client module on the specified interface to the DHCP server module on the default interface in the Trust zone. **Note:** On devices that can have multiple interfaces bound to the Trust zone, the default interface is the first interface bound to that zone and assigned an IP address. ■ **vendor** *id_str* Specifies the DHCP vendor by ID.

Examples: The following command configures interface **ethernet3** to perform automatic DHCP configuration after device power-up:

```
set interface ethernet3 dhcp client settings autoconfig
```

The following command enables (the forwarding of TCP/IP settings from the DHCP client module on the Untrust interface to the DHCP server module on the Trust zone interface):

```
set interface untrust dhcp client settings update-dhcpserver
```

dhcp relay

```
get interface interface dhcp relay
set interface interface dhcp relay { server-name name_str | service | vpn }
unset interface interface dhcp relay { server-name { name_str | ip_addr } | service | vpn }
```

dhcp relay	Configures the interface such that the security device can serve as a DHCP relay agent. ■ **server-name** *name_str* Defines the domain name of the external DHCP server from which the security device receives the IP addresses and TCP/IP settings that it relays to hosts on the LAN. ■ **service** Enables the security device to act as a DHCP server agent through the interface. ■ **vpn** Allows the DHCP communications to pass through a VPN tunnel. You must first set up a VPN tunnel between the security device and the external DHCP server.

The relay does not coexist with the DHCP server (OK with the client).

Example: The following command configures interface **ethernet4** to use an external DHCP server at IP address 1.1.1.10:

```
set interface ethernet4 dhcp relay server-name 1.1.1.10
```

dhcp server

set interface *interface* dhcp server { ... }
unset interface *interface* dhcp server { ... }

dhcp server — Makes the interface work as a DHCP server.

- **auto** Instructs the security device to check to see if there is a DHCP server already running on the network. If there is such a server, the DHCP server on the security device is disabled. If there is no DHCP server running on the network, the DHCP server on the security device is enabled. This is the default mode.
- **disable** Causes the DHCP server to always be off.
- **enable** Causes the DHCP server to always be on. The DHCP server on the security device always starts when the device is powered on.
- **ip** *ip_addr* { **mac** *mac_addr* | **to** *ip_addr* } Specifies either a specific IP address that is assigned to a host or the lower end of a range of IP addresses to use when the DHCP server is filling client requests.
  - **mac** This option allows you to statically assign an IP address to the host that is identified by the specified MAC address. The host is always assigned the specified IP address.
  - **to** Defines the upper end of a range of IP addresses to use when the DHCP server is filling client requests. The IP pool can support up to 255 IP addresses. The IP address must be in the same subnet as the interface IP or the DHCP gateway.
- **option** Specifies the DHCP server options for which you can define settings.
  - **dns1** *ip_addr* | **dns2** *ip_addr* | **dns3** *ip_addr* Defines the IP addresses of the primary, secondary, and tertiary Domain Name System (DNS) servers.
  - **gateway** *ip_addr* Defines the IP address of the gateway to be used by the clients. The IP address must be in the same subnet as the interface IP or the DHCP gateway.
  - **news** *ip_addr* Specifies the IP address of a news server to be used for receiving and storing postings for news groups.
  - **nis1** *ip_addr* | **nis2** *ip_addr* Defines the IP addresses of the primary and secondary NetInfo® servers, which provide the distribution of administrative data within a LAN.
  - **pop3** *ip_addr* Specifies the IP address of a Post Office Protocol version 3 (POP3) mail server.
  - **smtp** *ip_addr* Defines the IP address of a Simple Mail Transfer Protocol (SMTP) mail server.
  - **domainname** *name_str* Defines the registered domain name of the network.
  - **lease** *number* Defines the length of time, in minutes, for which an IP address supplied by the DHCP server is leased. For an unlimited lease, enter 0.
  - **netmask** *ip_addr* Defines the netmask of the gateway. The IP address must be in the same subnet as the interface IP or the DHCP gateway.
  - **nistag** *string* Defines the identifying tag used by the Apple® NetInfo database.
  - **wins1** *ip_addr* | **wins2** *ip_addr* Specifies the IP address of the primary and secondary Windows Internet Naming Service (WINS) servers.
- **service** Enables the security device to act as a DHCP server agent through the interface.

The server does not coexist with the DHCP relay (OK with the client).

Example: The following command configures the security device to act as a DHCP server agent through the interface **ethernet4**:

set interface ethernet4 dhcp server service

dhcp6 client

set interface *interface* dhcp6 client [...]
unset interface *interface* dhcp6 client [...]

dhcp6 client

Makes the interface work as a DHCPv6 client.

Note: This feature works only for IPv6-enabled interfaces. Before you can execute any of the options for this command, you must set up the interface in host mode, and activate the interface as a client by executing the command **set interface** *interface* **dhcp6 client**.

action set-default-route Directs the device to add a default route, to allow the client to access the IP address of the DHCPv6 server.

enable Causes the DHCPv6 client to always be active after device startup.

options Specifies the DHCPv6 client options for which you can define settings.

rapid-commit Allows the client to use an accelerated process to solicit information from the server. The client does this by using only two messages, a Solicit message (sent by the client) and a Reply message (received from the server).

Note: If the client only requests DNS configuration information, the exchanged messages are Information (sent by the client) and Reply (received from the server).

request (dns | pd | search-list } Specifies the type of information the client requests from the server.

dns Requests resolution of domain name to IP address.

pd Requests prefix delegation.

search-list Requests a list of the partial domain names for which the server performs domain name resolution.

pd [iapd-id *id_num* **]** Sets an association between the DHCPv6 client (a security device interface) and a preferred IPv6 prefix that a DHCPv6 server can delegate to the client.

The **iapd-id** *id_num* parameter sets the IAPD (Identity Association Prefix Delegation) number, which identifies a prefix (or a prefix pool) on the server. The client can use this value to request specific prefixes from the server. If you specify a nonzero IAPD value, it maps statically to a single prefix that has the same IAPD on the server. If you specify a zero IAPD value, or omit it (which assigns it a zero value by default) the IAPD maps it dynamically to a pool of prefixes on the server. The device treats all prefixes with a zero IAPD as belonging to this pool.

A Juniper Networks DHCPv6 server can contain up to 16 prefixes per client Device-Unique ID (DUID). For more information, see “dhcp6 server” on page 130.

prefix *ip_addr/pref_len number1* [*number2*] Specifies a preferred IPv6 prefix to receive from the DHCPv6 server.

number1 Specifies the preferred lifetime for the IPv6 prefix in the option (expressed in seconds). A value of 0 represents infinity.

number2 Specifies the recommended valid lifetime for the IPv6 prefix (expressed in seconds). A value of 0 represents infinity. If you omit the valid lifetime, the device sets the value to the preferred lifetime by default.

You can specify up to 16 preferred prefixes per client.

ra-interface *interface* sla-id *id_num1* sla-len *id_num2* Identifies a Router Advertisement (RA) interface. Through this interface, the device advertises the prefixes received from the DHCPv6 server. Local hosts receiving these RAs can use them to perform autoconfiguration.

The sla-id *id_num1* parameter identifies the SLA (Site-Local Aggregate) number, a hexadecimal value that denotes a subnet of the network connected to the RA interface. This value can be up to 64 bits in length.

The sla-len *id_num2* parameter specifies the length of the SLA (expressed in bytes). This value can range from 0 to 16.

If you specify a sla-id *id_num1* and sla-len *id_num2* of zero, the device automatically pads the SLA with 64 bits of zeroes.

prefer-server *id_num* Directs the DHCPv6 client to accept information only from the server that has the specified DUID (*id_num*).

Example: The following commands configure interface ethernet3 as a DHCPv6 client.

- Directs the device to request prefix delegation from the DHCPv6 server
- Specifies two preferred prefixes to request from the server:
  - Prefix 3eff:908e:729e::ef98:1/48, IAPD 100, preferred lifetime 120 seconds, valid lifetime 150 seconds
  - Prefix 3221:632f:a76f::3f22:1/48, IAPD 200, preferred lifetime 500 seconds, valid lifetime 600 seconds

```
set interface ethernet3 dhcp6 client
set interface ethernet3 dhcp6 client options request pd
set interface ethernet3 dhcp6 client pd iapd-id 100 prefix 3eff:908e:729e::/48 150
    120
set interface ethernet3 dhcp6 client pd iapd-id 200 prefix 3221:632f:a76f::/48 500
    600
save
```

dhcp6 server

set interface *interface* dhcp6 server { … }
unset interface *interface* dhcp6 server { … }

dhcp6 server

Makes the interface work as a DHCPv6 server.

DHCPv6 automatically provides configuration information from the DHCPv6 server to client hosts. For example, the server could provide DNS information, or allow automated delegation of long-lived IPv6 prefixes across an administrative boundary without knowing the topology of targeted local network.

Note: This feature only works for IPv6-enabled security devices. Before you can execute any of the options for this command, you must set up the interface for IPv6 in server mode, and activate the interface as a client by executing the command **set interface** *interface* **dhcp6 server**.

- **enable** Causes the DHCPv6 server to always be active after device startup.
- **options** Specifies the DHCPv6 server options for which you can define settings.
  - **client-duid duid** *string* [**name** *name_str*] Specifies a DUID (Device-Unique ID), which uniquely identifies a client host. Each DUID is based on the MAC address of the host, and uses the following format:

    00:03:00:01:xx:xx:xx:xx:xx:xx

    where:
    - 00:03: Indicates that this DUID is based on Link Layer address (DUID-LL).
    - 00:01: Indicates that the Link Layer address type is MAC address.
    - xx:xx:xx:xx:xx:xx: Contains the host MAC address.

    The optional **name** *name_str* setting assigns a user-friendly name to the DUID. You can use this name later when you create a prefix delegation using the **pd** option (described below).
  - **dns** { **dns1** *ip_addr* | **dns2** *ip_addr* | **dns3** *ip_addr* } Defines the IPv6 addresses of the primary, secondary, and tertiary Domain Name System (DNS) servers.
  - **pd** { **duid** *string* | **name** *name_str* } Defines how the device performs prefix delegation for a host device identified with a particular DUID. You can identify the DUID itself, or use its name associated with the DUID.
  - [**iapd-id** *id_num*] **prefix** *ip_addr*/*pref_len number1 number2* Specifies an IPv6 prefix to delegate to the host. If you specify an IAPD (Identity Association Prefix Delegation) number with the prefix, the device maps the prefix statically to that IAPD. A client can then specifically request the prefix using the IAPD. If you do not specify an IAPD, the device uses a default value of zero, and the prefix becomes part of a dynamic IPv6 prefix pool, available to any client request.

    Note: You can associate up to 16 prefixes for a client based on its DUID.

    - *number1* Specifies the preferred lifetime for the IPv6 prefix in the option (expressed in seconds). A value of 0 represents infinity.
    - *number2* Specifies the recommended valid lifetime for the IPv6 prefix (expressed in seconds). A value of 0 represents infinity. If you omit the valid lifetime, the device sets the value to the preferred lifetime by default.

  - **rapid-commit** Allows the server to use an accelerated process to transmit solicited information to the client. The client and server perform such solicitation by using only two messages, a Solicit message (received from the client) and a Reply message (sent by the server).

  Note: If the client only requests DNS configuration information, the exchanged messages are Information (received from the client) and Reply (sent by the server).

  - **search-list name** *name_str id_num* Makes an entry in the search list. This list which contains the partial domain names for which the server performs domain name resolution.

- **preference** *number* Enables the security device as a DHCP server agent through the interface, and assigns the server a preference number. The server sends this number to the client in ADVERTISE messages. The client uses this value to choose a specific server over others.

Example: The following commands configure interface ethernet3 as a DHCPv6 server and delegates two prefixes, making them available to a DHCPv6 client.

- Client DUID named Chris_Host, based on client MAC address aabbccddeeff
- Delegates the following prefixes:
- Prefix 3eff:908e:729e::ef98:1/48, IAPD 100, preferred lifetime 120 seconds, valid lifetime 150 seconds
- Prefix 3221:632f:a76f::3f22:1/48, IAPD 200, preferred lifetime 500 seconds, valid lifetime 600 seconds
- Because of the nonzero IAPD numbers, the server delegates both prefixes statically (not dynamically).

```
set interface ethernet3 dhcp6 server
set interface ethernet3 dhcp6 server options client-duid duid
    00:03:00:01:aa:bb:cc:dd:ee:ff name Chris_Host
set interface ethernet3 dhcp6 server options pd name Chris_Host iapd-id 100
    prefix 3eff:908e:729e::/48 150 120
set interface ethernet3 dhcp6 server options pd name Chris_Host iapd-id 200
    prefix 3221:632f:a76f::/48 500 600
```

For information about setting up the DHCPv6 client, see “dhcp6 client” on page 128.

dip

```
set interface interface dip id_num ip_addr1 [ ip_addr2 ] [ fix-port ]
set interface interface dip id_num ipv6 prefix ip_addr1/pref_len
set interface interface dip id_num shift-from ip_addr3
unset interface interface dip id_num
```

dip

Sets a Dynamic IP (DIP) pool. Each DIP pool consists of a range of addresses (IPv4 or IPv6). The security device can use the pool to dynamically allocate source addresses when the device applies Network Address Translation (NAT) to packets traversing the specified interface. This is necessary when performing the following tasks:

- Translating nonroutable local IP source addresses into routable addresses for outgoing packets.
- Translating IPv4 source addresses to IPv6 addresses. This allows the device to send IPv4 packets over an IPv6 network infrastructure.
- Translating IPv6 source addresses to IPv4 addresses. This allows the device to send IPv6 packets over an IPv4 network infrastructure.

The keywords and variables for the dip option are as follows:

- The ID number *id_num* identifies the DIP pool.
- The first IP address (*ip_addr1*) represents the start of the IP address range. (A DIP pool can consist of a single IP address, or many.) The second IP address (*ip_addr2*) represents the end of the IP address range.
- The ipv6 prefix *ip_addr*/*pref_len* option specifies an address range of an IPv6 subnet, expressed as an IPv6 address (*ip_addr*) and prefix length.
- shift-from *ip_addr3* Defines a one-to-one mapping from an original source IP address to a translated source IP address for a range of IP addresses starting from *ip_addr3*. Such a mapping ensures that the security device always translates a particular source IP address from within that range to the same translated address within a DIP pool.

Be sure to exclude the following IP addresses from a DIP pool:

- The WebUI management IP address
- The interface and gateway IP addresses
- Any Virtual IP (VIP) and Mapped IP (MIP) addresses

Example 1: The following commands allow local IPv6 hosts to send service requests over an IPv4 WAN infrastructure. The security device uses an IPv4 DIP pool to dynamically allocate IPv4 source addresses to packets sent from the local IPv6 hosts to remote IPv6 hosts through a peer security device.

- Interface ethernet1 in IPv6 router mode
- Global aggregatable IPv6 address 2e10::530:22:aaff:fe23/64
- DIP ID number 10
- DIP IPv4 address range from `2.1.10.2 through 2.1.10.25`

```
set interface ethernet1 zone trust
set interface ethernet1 ipv6 mode router
set interface ethernet1 ipv6 enable
set interface ethernet1 ipv6 ip 2e10::530:22:aaff:fe23/64
set interface ethernet1 ipv6 ra transmit
set interface ethernet3 zone untrust
set interface ethernet3 ip 2.1.10.1/24
set interface ethernet3 dip 10 2.1.10.2 2.1.10.25
```

The following commands define local and remote subnets, and a NAT policy that allows outgoing HTTP requests from the local subnet. When this policy is active, the security device dynamically generates IPv4 source addresses from the DIP pool for outgoing packets.

```
set address trust Local_Hosts 2e10::530:22:aaff:fe0/64
set address untrust Remote_Hosts 322::49a:3af:c4ff:fe70/64
set policy from trust to untrust Local_Hosts Remote_Hosts http nat dip 10 permit
```

Example 2: The following commands allow local IPv4 hosts to communicate over an IPv6 WAN infrastructure. The security device uses an IPv6 DIP pool to dynamically allocate IPv6 source addresses to packets sent from local IPv4 hosts to remote IPv4 hosts through a peer security device.

- Trust zone interface ethernet1 address range 10.1.1.0/24
- Untrust zone interface ethernet3 set to IPv6 host mode
- Untrust zone interface ethernet3 global aggregatable IPv6 address `3ff1::450:af:34ff:fe77/64`
- DIP ID number 5, with address range from 3ff1::450:af:34ff:fe1 through 3ff1::450:af:34ff:fe50

```
set interface ethernet1 zone trust
set interface ethernet1 ip 10.1.1.0/24
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode host
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 3ff1::450:af:34ff:fe01/64
set interface ethernet3 dip 5 3ff1::450:af:34ff:fe10 3ff1::450:af:34ff:fe50
```

A policy can use the DIP ID and NAT to generate IPv6 source addresses dynamically from the DIP pool. For example, the following commands define local and remote subnets, and a NAT policy that allows outgoing HTTP requests from the local subnet. When this policy is active, the security device dynamically generates IPv6 source addresses from the DIP pool for outgoing packets.

```
set address trust Local_Hosts 10.1.1.7/24
set address untrust Remote_Hosts 10.10.2.12/24
set policy from trust to untrust Local_Hosts Remote_Hosts http nat dip 5 permit
```

enable (ipv6)

set interface *interface* ipv6 enable

enable

Enables the specified interface for IPv6.

Note: Before you can use this command, you must perform the following actions:

- Enable the device for IPv6 by executing set env ipv6 = enable.
- Set the interface in host mode or router mode by executing the following command:

  set interface *interface* ipv6 mode { host | router }

ext ip

set interface *interface* ext ip *ip_addr/mask* dip *number* { ... }
unset interface *interface* ext ip *ip_addr/mask* dip *number*

ext ip

The ext ip *ip_addr* option is a dual-stacked (IPv4/IPv6) command that configures a DIP in a different subnet from the interface's subnet. For example, an interface could have IP address 1.2.10.1/24, and the extended DIP could be 2.2.3.1/24.

- dip *id_num* Sets a Dynamic IP (DIP) pool. See "dip" on page 132.
- fix-port Keeps the original source port number in the packet header. Does not apply the Port Address Translation (PAT).

Example: The following command creates an address (1.1.100.110) in a DIP (ID 10) for interface ethernet3 (IP address 10.1.10.10):

```
set interface ethernet3 ext ip 10.1.10.10/24 dip 10 1.1.100.110
```

gateway

set interface *interface* gateway *ip_addr* [no-default-route]
unset interface *interface* gateway

gateway

The IP address for the default gateway to which the security device forwards packets that are destined for networks beyond the immediate subnet of the specified interface. The no-default-route switch specifies that there is no default route for this gateway.

Example: The following command specifies the IP address of a remote gateway peer (1.1.10.10) for the ethernet4 interface:

```
set interface ethernet4 gateway 1.1.10.10
```

inverse-nd (ipv6)

set interface *interface* ipv6 inverse-nd
set interface *interface* ipv6 inverse-nd link-mtu

ipv6 inverse-nd — Enables inverse Neighbor Discovery (ND) on the Frame Relay interface. This option is disabled by default.

link-mtu — Defines the maximum size (in bytes) of any IPv6 packet sent by a host over a link. The inverse-nd packet carries the maximum size of the link and updates the maximum transmission unit (mtu) of the peer on the other end of the link.

Example: The following command displays inverse ND information about interface *serial0.1*:

get interface serial0.1 inverse-nd

interface-id (ipv6)

set interface *interface* ipv6 interface-id *id_num*

interface-id — (ipv6 interfaces only.) Sets the EUI identification for the interface. The EUI is a 64-bit hexadecimal extension of the ethernet MAC address. The NetScreeen device uses this value to autoconfigure an IPv6 link-local IP address for the interface.

Note: Although zero-substitution notation (::) is allowed in IPv6 addresses, it is not allowed anywhere in the EUI value.

ip

set interface *interface* ip *ip_addr/mask* [secondary]
unset interface *interface* ip *ip_addr*

ip — The IP address *ip_addr* and netmask *mask* for the specified interface or subinterface. The secondary switch specifies that the IP address is a secondary address.

Example: The following commands create logical interface ethernet3/1.2, bind it to the Trust zone, and assign it IP address 10.1.40.3/24:

set interface ethernet3/1.2 zone trust
set interface ethernet3/1.2 ip 10.1.40.3/24

ip (IPv6)

set interface *interface* **ipv6 ip** *ip_addr/prf_len*

enable — (ipv6 interfaces only.) Sets an IPv6 unicast address and prefix for the interface.

Example: The following command assigns IPv6 unicast address fe80::200:ff:fe00:8, with a prefix length of 60, to the ethernet4 interface:

set interface ethernet4 ipv6 ip fe80::200:ff:fe00:8/60

loopback-group

set interface *interface1* loopback-group loopback.*n* unset interface *interface1* loopback-group loopback.*n*

loopback-group Adds a specified interface (*interface1*) to the loopback group for a designated loopback interface (loopback.*n*). All members in the loopback group can share the MIP (Mapped IP) and DIP (Dynamic IP) definitions assigned to the loopback interface itself.

Example: The following commands add interfaces ethernet1 and ethernet2 to the loopback group for loopback.1, and then assign a MIP to loopback.1. This allows both ethernet1 and ethernet2 to use the assigned MIP.

```
set interface ethernet1 loopback-group loopback.1
set interface ethernet2 loopback-group loopback.1
set int loopback.1 mip 1.1.1.1 host 10.1.1.8 netmask 255.255.255.0
```

manage

set interface *interface* manage
    { ident-reset **|** nsmgmt **|** ping **|** snmp **|** ssh **|** ssl **|** telnet **|** web }
unset interface *interface* manage
    { ident-reset **|** nsmgmt **|** ping **|** snmp **|** ssh **|** ssl **|** telnet **|** web }

manage Enables or disables monitoring and management capability through the interface.

- ident-reset Directs the security device to send a TCP Reset announcement, in response to an IDENT request, to port 113.
- nsmgmt Enables or disables NetScreen-Security Manager (NSM) on the interface. NSM is an enterprise-level management application that configures security devices from remote hosts. For more information, see “nsmgmt” on page 170.
- ping Enables (or disables) ping through the interface.
- snmp Enables (or disables) SNMP management through the interface.
- ssh Enables (or disables) SSH management through the interface.
- ssl Enables (or disables) SSL management through the interface.
- telnet Enables (or disables) Telnet management through the interface.
- web Enables (or disables) web management through the interface.

Example: The following command enables management of SSH through interface ethernet3:

```
set interface ethernet3 manage ssh
```

manage-ip

set interface *interface* manage-ip *ip_addr*
unset interface *interface* manage-ip

manage-ip Defines the Manage IP address for the specified physical interface. External applications such as Telnet or WebUI can use this address to configure and monitor the security device. (This address must be in the same subnet as the interface IP address.)

Example: The following commands bind interface ethernet4/1 to the Trust zone, then set the Manage IP address to 10.1.10.10:

set interface ethernet4/1 zone trust
set interface ethernet4/1 manage-ip 10.1.10.10

mip

set interface interface mip *ip_addr* host *ip_addr* [netmask mask]
set interface interface mip *ip_addr* ipv6 host *ip_addr* [netmask mask]
set interface interface mip *ip_addr* ipv6 prefix *ip_addr/pref_len* [netmask mask]
set interface interface mip *ip_addr/mask* ipv6 ipv4
set interface interface mip ip_addr/*mask* prefix *ip_addr/pref_len*

mip

Defines a Mapped IP (MIP) address for the interface. The security device directs traffic sent to the MIP (*ip_addr1*) to the host with the IPv4 or IPv6 address *ip_addr*. Setting a MIP for an interface in any zone generates a book entry for the MIP in the Global zone address book. The Global zone address book keeps all the MIPs of all interfaces, regardless of the zone to which the interfaces belong. You can use these MIP addresses as the destination addresses in policies between any two zones, and as the source addresses when defining a policy from the Global zone to any other zone.

- host *ip_addr2* Specifies the IP address of a host device that uses IPv4 addressing. The netmask value specifies either a single one-to-one mapping or a mapping of one IP address range to another.

  Note: Exclude the interface and gateway IP addresses, and any Virtual IP addresses in the subnet from the MIP address range.)

- ipv6 Specifies the IP address of a host device that uses IPv6 addressing.
  - host *ip_addr* Specifies the IPv6 address of a host device that uses IPv6 addressing.
  - prefix *ip_addr/pref_len* Specifies a range of address for host devices that use IPv6 addressing.
  - ipv4 Instructs the device to directly translate IPv6 addresses into IPv4-mapped addresses. An IPv4-mapped address is a special global unicast IPv6 address containing an embedded IPv4 address. security devices use such addresses when they need to send IP traffic over an IPv6/IPv4 border to a remote IPv4 network. For example, when an IPv6 host transmits a service request packet through a security device into an IPv4 network, the device represents each IPv4 destination host using an IPv4-mapped address.

    It is customary to express IPv4-mapped addresses in the format IPv6::a.b.c.d, where IPv6 is the IPv6 subnet portion of the address, and a.b.c.d is the embedded IPv4 address (in decimal notation). However, the IPv4 address is actually in hexadecmal format, and resides in the last 32 bits (four octets) of the IPv4-mapped address.
- vrouter *vrouter* Identifies the virtual router containing a route to the host device.

Example 1: The following commands use a MIP to allow remote hosts to request HTTP services from a local HTTP server, located in a nonroutable subnet, over a public IPv4 WAN infrastructure. The MIP directly translates all outgoing source IP addresses into public addresses.

1. Set up ethernet interfaces.

```
unset interface ethernet2 ip
unset interface ethernet2 zone
unset interface ethernet3 ip
unset interface ethernet3 zone

set interface ethernet2 zone trust
set interface ethernet2 ip 10.100.2.1/24
set interface ethernet3 zone untrust
set interface ethernet3 ip 1.1.12.1/24
```

2. Create a MIP definition for the interface bound to the Untrust zone.

```
set interface ethernet3 mip 2.2.22.5 host 10.100.2.5 vrouter trust-vr
```

3. Create a policy definition that invokes the MIP.

```
set policy from untrust to trust any mip(2.2.22.5) http nat permit
save
```

Example 2: The following commands create a policy that uses a MIP, allowing remote hosts to request HTTP services from a local IPv4 HTTP server over an IPv6 WAN backbone. The MIP statically translates any incoming IPv6 address into the local server's IPv4 address.

- Ethernet1 interface bound to trust zone, with IPv4 address 10.100.2.150/24
- Local server IP address 10.1.2.155
- Ethernet3 interface bound to untrust zone, with IPv6 address 32f1::250:af:34ff:fe10/64
- IPv6 address 32f1::250:af:34ff:fe5 mapped to local server IPv4 address 10.1.2.155

1. Set up ethernet interfaces:

```
set interface ethernet1 zone trust
set interface ethernet1 ip 10.100.2.150/24
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode host
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 32f1::250:af:34ff:fe10/64
```

2. Create a MIP definition for the interface bound to the Untrust zone:

```
set interface ethernet3 mip 32f1::250:af:34ff:fe5 ipv6 host 10.100.2.155
    vrouter trust-vr
```

3. Create a policy definition that invokes the MIP:

```
set policy from untrust to trust any-ipv6 mip(32f1::250:af:34ff:fe5)
    http nat permit
save
```

mode (ipv6)

set interface *interface* ipv6 mode host
set interface *interface* ipv6 mode router

mode

(ipv6 interfaces only.) Starts an IPv6 instance for the interface. This instance gives the interface IPv6 capability, either as a host or as a router.

- host Places the IPv6 instance in Host mode and broadcasts a Router Solicitation message to discover and identify routers to use as gateways.
- router Place the IPv6 instance in Router mode and broadcasts a Router Advertisement message to announce its availability to other devices as a default router.

mtu

set interface *interface* mtu *number*
unset interface *interface* mtu

mtu

Sets the Maximum Transmission Unit (MTU) for the interface. The MTU is the largest physical packet size (in octets) that the device can transmit on the interface. The security device must fragment any messages larger than the MTU before sending them. The default MTU size is 1500 octets. Enter a value between 1280 and 8192.

The range is from 1200 through 4080 octets when IPv6 is enabled on the Frame Relay interface.

nat

set interface *interface* nat

nat

Directs the device to perform Network Address Translation (NAT) on outbound traffic from the trusted LAN. This option is available only when the device is in Route mode, in which the interfaces have assigned IP addresses.

nd (ipv6)

set interface *interface* ipv6 nd { ... }

nd

(ipv6 interfaces only.) Configures the IPv6 interface instance for neighbor discovery.

- **base-reachable-time** *number* Specifies the *approximate* length of time (expressed in seconds) that the security device maintains Reachable status for a neighbor after the security device transmits a Neighbor Solicitation (NS) message to the neighbor and receives a Neighbor Advertisement (NA) message in reply. The default is 30 seconds, and the maximum is 36,000 seconds. (If the interface accepts router advertisements, any received advertisements override this setting.)

  Note: The Base Reachable Time setting only approximates the actual time that the status remains Reachable. The exact time interval is called Reachable Time. The security device determines Reachable Time randomly, using the Base Reachable Time as a baseline value. The resulting Reachable Time is usually within 50 to 150 percent of the Base Reachable Time setting.
- **dad-count** Specifies the Duplicate Address Detection (DAD) retry count, which is the number of consecutive NS messages that the security device sends while performing DAD for the interface. A setting of 0 disables DAD for the interface. Valid settings are from 0 to 10, and the default setting is 3. When the device uses DAD, it tests the IPv6 address for uniqueness. The security device does not assign the interface any address found to be a duplicate. The device only applies DAD to newly assigned addresses, or when the interface goes down and comes back up again.
- **nud** Enables Neighbor Unreachability Detection (NUD), which identifies failure of a communication path to a neighbor device. The security device stores reachability states for each neighbor in a Neighbor Cache table. Each entry in this table tracks the current reachability status of a neighbor.
- **probe-time** The time interval (in seconds) between retransmissions of NS messages after the security device determines that the neighbor is unreachable. The default setting is 10 seconds, and the maximum value is 300 seconds.

  The security device stores reachability states for each neighbor in a Neighbor Cache table. Each entry in this table tracks the current reachability status of a neighbor. While an entry status is Incomplete or Probe Forever (as when the neighbor is a next-hop gateway), the security device attempts to confirm the reachability of the neighbor. Each attempt is called a probe. During a probe, the security device transmits NS requests to the neighbor. The Probe Time setting specifies the interval of time (expressed in seconds) between probes.
- **retransmit-time** The time interval (in seconds) between retransmissions of NS messages during a probe operation. The default setting is 1 second.

nhtb

set interface interface.*number* nhtb *ip_addr* vpn *tunn_str*
unset interface interface.*number* nhtb *ip_addr*

nhtb	Binds the specified VPN tunnel (vpn *tunn_str*) to the tunnel interface and manually maps the specified VPN tunnel to the IPv4 or IPv6 address of a remote peer's tunnel interface (*ip_addr*) in the Next-Hop Tunnel Binding (NHTB) table. After that, you can enter a static route in the route table that uses that tunnel interface IP address as the gateway.

Example: With the following commands, you first bind vpn1 to tunnel.1 and map vpn1 to 10.2.3.1, which is the IP address of the remote peer's tunnel interface. Then you define a static route to 10.2.2.0/24, which is the address of the remote peer's internal LAN, through tunnel.1 in the trust-vr routing domain, using the remote peer's tunnel interface IP address (10.2.3.1) as the next-hop gateway:

```
set interface tunnel.1 nhtb 10.2.3.1 vpn vpn1
set vrouter trust-vr route 10.2.2.0/24 interface tunnel.1 gateway 10.2.3.1
```

ntp-server

set interface *interface* ntp-server
unset interface *interface* ntp-server

ntp-server	Enables NTP service on a specified Layer 3 interface with an IP address. If this option is set, the security device acts as an NTP server serving requests from the subnet peers. Note: Currently, ScreenOS supports only Unicast mode.

phy

set interface *interface* phy { ... }
unset interface *interface* phy { ... }

phy — Defines the physical connection mode on the specified interface.

- auto The security device automatically decides whether to operate at full-duplex or half-duplex (as required by the network device to which it is connected).
- full Forces the security device to operate at full-duplex. Specify either 100Mbps or 10Mbps.
- half Forces the security device to operate at half-duplex. Specify either 100Mbps or 10Mbps.
- holddown *number* Sets the holddown time for the link, in increments of 100 milliseconds.
- link-down Forces the physical link down.
- manual Specifies Manual mode for a gigabit link.

protocol

get interface *interface* protocol { ... }
set interface *interface* protocol { ... }
unset interface *interface* protocol { ... }

protocol { rip | ripng }

Sets, unsets, or displays the current routing protocol settings for the interface.

Note: If the interface uses IPv4 addressing, use Routing Information Protocol (RIP). If the interface uses IPv6 addressing, use Routing Information Protocol next-generation (RIP-ng). You cannot view the RIPng options until you enter the vrouter context and set the protocol to **RIPng**.

- **route-map** *name_str* Specifies the route-map on which to filter incoming routes (routes learned by RIP/RIPng) or outgoing routes (routes advertised by RIP/RIPng).
  - **in** Specifies the route map is to be used for incoming routes.
  - **out** Specifies the route map is to be used for outgoing routes.
- **authentication** { **password** *pswd_str* | **md5** *key_str* **key-id** *id_num* } Specifies the authentication method used to verify RIP/RIPng neighbors.
  - **password** specifies a clear-text password used for verification. If you specify password authentication, you must also specify an 8-byte password.
  - **md5** directs the security device to use the Message Digest version 5 (MD5) authentication algorithm for verification. If you specify MD5 authentication, you must also specify a 16-byte key and, optionally, a key identifier (the default identifier is 0). You can specify more than one MD5 key with different key identifier numbers (between 0-255). If there are multiple MD5 keys configured, you can use the **active-md5-key-id** option to select the key identifier of the key to be used for authentication.
- **enable** Enables RIP/RIPng on the specified interface.
- **metric** *number* Configures the RIP/RIPng metric for the specified interface. The default metric is 1.
- **passive-mode** Specifies that the interface is to receive but not transmit RIP/RIPng packets.
- **split-horizon** Enables the split-horizon function on the specified interface. If **split-horizon** is enabled, RIP/RIPng does not advertise routes learned from a neighbor back to the same neighbor. This avoids the routing-loop problem that occurs in some routing situations. If **split-horizon** is disabled, RIP/RIPng advertises routes learned from a neighbor back to the same neighbor with a metric of 16. By default, **split-horizon** is enabled.

  When you enable the **poison-reverse** switch, RIP/RIPng still advertises routes learned from a neighbor back to the same neighbor, but defines the metric for those routes as infinity (16). This causes the neighbor to immediately remove the route, thus breaking a potential routing loop faster than with **split-horizon** alone. When you disable this switch, RIP/RIPng advertises routes learned from a neighbor back to the same neighbor with the correct metric.

protocol ospf

Sets, unsets or displays the current routing protocol settings for the interface.

- **area** { *ip_addr* | *number* } Assigns the interface to the specified OSPF area. OSPF areas divide the internetwork into smaller, more manageable constituent pieces. This technique reduces the amount of information that each router must store and maintain about all the other routers.

- authentication { md5 *key_str* [key-id *id_num*] | password *pswd_str* } Specifies the authentication method, including MD5 key string, the key identifier number (the default is 0), and password. You can specify more than one MD5 key with different key identifier numbers (between 0-255). If there are multiple MD5 keys configured, you can use the active-md5-key-id option to select the key identifier of the key to be used for authentication.

- cost *number* Specifies the desirability of the path associated with the interface. The lower the value of this metric, the more desirable the interface path.

- dead-interval *number* Specifies the maximum amount of time that the security device waits, after it stops receiving packets from the neighbor, before classifying the neighbor as offline.

- disable Disables OSPF on the interface, thus preventing transmission or receipt of OSPF packets through the interface.

- hello-interval *number* Specifies the amount of time in seconds that elapse between instances of the interface sending Hello packets to the network announcing the presence of the interface.

- link-type Configures the interface link type. By default, an ethernet interfaces is treated as an interface to a broadcast network with multiple attached routers. For broadcast networks, the Hello protocol elects a Designated Router and Backup Designated Router for the network.

  - point-to-point Configures the interface as a point-to-point link.

- neighbor-list *number1* [*number2* [*number3* [*number4*]]] Specifies the number of access lists (up to four), from which the local virtual router accepts valid neighbors to form adjacencies. The access list must be in the virtual router to which the interface is bound.

- passive Specifies that the IP address of the interface is advertised into the OSPF domain as an OSPF route and not as an external route, but the interface does not transmit or receive OSPF packets. This option is useful when BGP is also enabled on the interface.

- priority *number* Specifies the router election priority.

- retransmit-interval *number* Specifies the amount of time (in seconds) that elapses before the interface resends a packet to a neighbor that did not acknowledge a previous transmission attempt for the same packet.

- transit-delay *number* Specifies the amount of time (in seconds) that elapses before the security device advertises a packet received on the interface.

Example: In the following IPv6 example, you set the untrust-vr to use RIPng, you enable the instance of RIPng on the untrust-vr, and then you enable RIPng on the ethernet3 interface.

```
device-> set vrouter untrust-vr
device(untrust-vr)-> set protocol ripng
device(untrust-vr/ripng)-> set enable
device(untrust-vr/ripng)-> exit
device(untrust-vr)-> exit
device-> set interface ethernet3 protocol ripng
device-> set interface ethernet3 protocol ripng enable
device-> save
```

ra (ipv6)

get interface *interface* ipv6 ra
set interface *interface* ipv6 ra { ... }

ra

(ipv6 interfaces only.) Sets the parameters for IPv6 router advertisements transmitted by the interface to on-link IPv6 hosts. This option is available in Route mode (IPv6) only.

A Juniper Networks interface (configured to operate as an IPv6 router) uses these parameters to enable autoconfiguration for on-link IPv6 hosts. Host autoconfiguration eliminates (or reduces) the need to manually assign addresses to individual hosts on the IPv6 link. Because the autoconfiguration does not require a stateful configuration protocol such as DHCPv6, it is said to be *stateless*.

- accept When enabled, directs the security device to learn of the existence and identity of IPv6 routers by accepting Router Advertisement (RA) messages.

  Note: This option is available only when the interface is in host mode. It is not available when the interface is in router mode.

- hop-limit *number* Specifies the hop limit for packets transmitted by any local IPv6 host that uses RAs from the IPv6 router for address autoconfiguration. Setting the Current Hop Limit value to zero denotes an unspecified number of hops.

- link-mtu Instructs the IPv6 router to advertise the link-Maximum Transmission Unit (link-MTU) in router advertisements. The link-MTU is the maximum size (expressed in bytes) of any unfragmented IPv6 packet that can traverse the IPv6 link. (The default link-MTU is 1500 bytes for Ethernet and 1490 for PPPoE. In all cases, the link-MTU must be in the range of 1280 and 1500.)

- link-address Enables the Link Layer Address flag, which directs the IPv6 router to include the Link Layer (MAC) address of the router in outgoing RA messages.

- default-life-time *number* Specifies the interval of time (expressed in seconds) that hosts may consider the IPv6 router to be the default router after receiving the last transmitted RA.

- managed Enables the Managed Configuration Flag (M flag), which directs the local IPv6 hosts to use a stateful address autoconfiguration protocol, such as DHCPv6, to generate host addresses.

  Note: For local IPv6 hosts to perform stateless address autoconfiguration, you must disable this setting.

- **max-adv-int** Specifies the maximum time interval allowed between transmission of unsolicited multicast RAs from the IPv6 router.
- **min-adv-int** Specifies the minimum time interval allowed between transmission of unsolicited multicast RAs from the IPv6 router.
- **other** Enables the Other Stateful Configuration flag (O flag), which directs the local IPv6 hosts to use a stateful address autoconfiguration protocol, such as DHCPv6, to configure any parameters other than host addresses.

  Note: For local IPv6 hosts to perform stateless address autoconfiguration, you must disable this setting.
- **preference** Specifies the preference level for the IPv6 router (**high**, **medium**, or **low**). When a host receives the RA from the router, and other routers are present, the preference level determines whether the host views the router as a primary or secondary router.
- **prefix** *ip_addr*/*pref_len number1 number2* Specifies a prefix that the receiving hosts on the same link can use to autoconfigure themselves. The prefix consists of a valid IPv6 address and prefix length, as with 3ffe::200:1234:7/64.

  Note: The security device advertises global prefixes, never link-local prefixes.
  - **autonomous** Instructs the host receiving the router advertisement to use stateless configuration to generate addresses from the specified prefix.
  - **onlink** Instructs the host receiving the router advertisement that the specified prefix is onlink and should should use the prefix for stateless configuration.

  The *number1* parameter specifies the valid lifetime of the prefix in seconds. The *number2* parameter specifies the preferred lifetime of the prefix in seconds.
- **reachable-time** Specifies the length of time (in seconds) that a host may consider the security device reachable after receiving a transmitted RA. When a security device transmits a Neighbor Solicitation (NS) message to a neighbor and receives a Neighbor Advertisement (NA) message in reply, the device sets the neighbor reachability status to **Reachable**. The Base Reachable Time setting specifies the approximate length of time (expressed in seconds) that the security device maintains the Reachable status. After this time interval passes, the status goes to Stale mode.

  Note: The NUD algorithm generates this value from the Base Reachable Time. For more information, see “nd (ipv6)” on page 140.

  The default is 0 seconds, and the maximum is 3,600 seconds. (If the interface is in host mode and the accept flag is enabled, any received advertisements override this setting.)
- **retrans-time** Instructs the security device to include the Retransmission Time interval in outgoing RA messages. This interval (expressed in seconds) is the time that elapses between retransmissions of NS messages.
- **transmit** Allows the security device to send RA messages to on-link hosts. After you enable this setting, the IPv6 router immediately broadcasts a route advertisement to the hosts in the link. In addition, the device broadcasts an RA automatically when it receives a RS from a host, or when you change any RA setting on the router.

  Note: For local IPv6 hosts to perform stateless address autoconfiguration, you must enable this setting.

route

set interface *interface* route

route	Directs the device to run in Route mode, in which the interfaces have assigned IP addresses.

route-deny

set interface *interface* route-deny
unset interface *interface* route-deny

route-deny	Enabling this flag blocks all traffic in or out of the same interface. This includes traffic between the primary subnet and any secondary subnet, and one secondary subnet to another secondary subnet.

screen

get interface *interface* screen

screen	Displays the current firewall (screen) counters.

secondary

set interface *interface* ip *ip_addr/mask* secondary
get interface *interface* secondary [*ip_addr*]

secondary	Sets or displays the secondary address configured for the interface.

tag

set interface *interface.n* tag *id_num* zone *zone*

tag	Specifies a VLAN tag (*id_num*) for a virtual (logical) subinterface. The interface name is interface.n, where *n* is an ID number that identifies the subinterface. For more information, see “Interfaces” on page 319.

Example: The following command creates a subinterface for physical interface ethernet3/1, assigns it VLAN tag 300, and binds it to the Untrust zone:

set interface ethernet3/1.2 tag 300 zone untrust

track-ip

```
get interface interface track-ip
set interface interface track-ip
    [
    dynamic |
    threshold number |
    ip ip_addr
        [
        interval number |
        threshold number |
        weight number
        ]
    ]
unset interface interface track-ip
    [
    dynamic |
    ip ip_addr [ interval | threshold | weight ] |
    threshold
    ]
```

track-ip

Sets, unsets, or displays the tracking of IP addresses for the specified interface.

- dynamic Configures tracking of the IP address of the default gateway for the interface.
- threshold *number* Specifies the failure threshold for the interface. If the weighted sum of all tracked IP failures on the interface is equal to or greater than the threshold, the interface is considered down and failover to the backup occurs. Unsetting the tracked IP threshold on the interface sets the threshold to the default value of 1.
- ip *ip_addr* Configures tracking for the specified IP address. You can specify the following options:
    - interval *number* Specifies the interval, in seconds, that ping requests are sent to the tracked IP address. If you are unsetting the interval for the tracked IP address, the interval is changed to the default value of 1.
    - threshold *number* Specifies the failure threshold for the tracked IP address. If the number of consecutive ping failures to the tracked IP address is equal to or greater than the threshold, the tracked IP address is considered failed. If you are unsetting the threshold for the tracked IP address, the device changes the threshold to the default value (3).
    - weight *number* Specifies the weight associated with the failure of the tracked IP address. If a tracked IP address fails, its weight is used to calculated the weighted sum of all tracked IP failures on the interface. If you are unsetting the weight for the tracked IP address, the weight is changed to the default value of 1.

Examples: The following command defines IP tracking for an interface.

- IP address 1.1.1.1 on the ethernet3 interface
- Ping interval of 10 seconds
- Tracked IP address failure threshold of 5

```
set interface ethernet3 track-ip ip 1.1.1.1 interval 10 threshold 5
```

The following command sets the tracking threshold for the ethernet3 interface to 3:

```
set interface ethernet3 track-ip threshold 3
```

tunnel

set interface tunnel.*n* { zone *name_str* | protocol { bgp | ospf | rip } }

tunnel.*n*

Specifies a tunnel interface. The *n* parameter is an ID number that identifies the tunnel interface.

Note: Use the IP option only after adding the tunnel to a specific zone.

Example: The following commands create a tunnel interface named tunnel.2 with IP address 172.10.10.5/24:

```
set interface tunnel.2 zone untrust
set interface tunnel.2 ip 172.10.10.5/24
```

vip

set interface *interface* vip [*ip_addr* | *interface-ip*]
    [+] *port_num* [*name_str ip_addr* [manual]]

vip

Defines a Virtual IP (VIP) address for the interface by using the IP address of the interface (*interface-ip*) or any IP address (*ip_addr*) on the interface so you can map routable IP addresses to internal servers and access their services.

The *port_num* parameter is the port number, which specifies which service to access. The *name_str* and *ip_addr* parameters specify the service name and the IP address of the server providing the service, respectively. The manual switch turns off server auto detection. Using the + operator adds another service to the VIP.

Example: The following command creates a VIP for interface ethernet3, specifying the MAIL service (ID 25):

```
set interface ethernet3 vip 1.1.14.15 25 MAIL 10.1.10.10
```

vlan trunk

set interface vlan1 vlan trunk
unset interface vlan1 vlan trunk

vlan trunk

(vlan1 interface only.) Determines whether the security device accepts or drops Layer 2 frames. The device makes this decision only when the following conditions apply:

- The security device is in Transparent mode.
- The device receives VLAN tagged frames on an interface.

The device then performs one of two actions:

- Drops the frames because they have tags
- Ignores the tags and forwards the frames according to MAC addresses

The vlan trunk interface switch determines which action the device performs. For example, the command set interface vlan1 vlan trunk instructs the security device to ignore the tags and forward the frames. This action closely follows that of a Layer 2 switch trunk port.

webauth

set interface *interface* webauth

webauth	Enables WebAuth user authentication.

webauth-ip

set interface *interface* webauth-ip *ip_addr*

webauth-ip	Specifies the WebAuth server IP address for user authentication. Before sending service requests (such as MAIL) through the interface, the user must first browse to the WebAuth address with a web browser. The security device presents a login screen, prompting for username and password. After successfully entering the username and password, the user can send service requests through the interface. To protect an interface with the WebAuth feature, you must create a security policy with the set policy command, specifying the webauth switch. To specify the WebAuth server, use the set webauth command.

zone

set interface *interface* zone *zone*
unset interface *interface* zone

zone	Binds the interface to a security zone.

Example: The following command binds interface ethernet2/2 to the Trust zone:

set interface ethernet2/2 zone trust

ip

Use the ip commands to set or display IP parameters for communication with a TFTP server.

A security device can use TFTP servers to save or import external files. These files can contain configuration settings, software versions, public keys, error messages, certificates, and other items.

Syntax

get

get ip tftp

set

```
set ip tftp
    {
    retry number |
    timeout number
    }
```

Keywords and Variables

retry

```
set ip tftp retry number
```

retry	The number of times to retry a TFTP communication before the security device ends the attempt and generates an error message.

Example: The following command sets the number of retries to 7:

```
set ip tftp retry 7
```

timeout

```
set ip tftp timeout number
```

timeout	Determines how the long (in seconds) the security device waits before terminating an inactive TFTP connection.

Example: The following command sets the timeout period to 15 seconds:

```
set ip tftp timeout 15
```

ip-classification

Use the **ip-classification** command to display the current IP-based traffic classification.

IP-based traffic classification allows you to use virtual systems without VLANs. Instead of VLAN tags, the security device uses IP addresses to sort traffic, associating a subnet or range of IP addresses with a particular system (root or vsys).

Using IP-based traffic classification exclusively to sort traffic, all systems share the following:

- The untrust-vr and a user-defined internal-vr
- The Untrust zone and a user-defined internal zone
- An Untrust zone interface and a user-defined internal zone interface

To designate a subnet or range of IP addresses to the root system or to a previously created virtual system, you must issue one of the following CLI commands at the root level:

```
set zone zone ip-classification net ip_addr/mask { root | vsys name_str }
set zone zone ip-classification range ip_addr1-ip_addr2 { root | vsys name_str }
```

For more information, see “zone” on page 309.

Syntax

```
get ip-classification [ zone zone ]
```

Keywords and Variables

zone

get ip-classification zone *zone* [ip *ip_addr*]

zone	The name of the security zone. The ip *ip_addr* option specifies a particular address in a specific zone.

Example: To display the current IP classification for the ethernet1 zone:

get ip-classification zone untrust

ippool

Use the ippool commands to associate the name of an IP pool with a range of IP addresses. The security device uses IP pools when it assigns addresses to dialup users using Layer 2 Tunneling Protocol (L2TP).

Syntax

get

get ippool *name_str*

set

set ippool string *ip_addr1 ip_addr2*

Keywords and Variables

Variables

get ippool *name_str*
set ippool string *ip_addr1 ip_addr2*
unset ippool *name_str*

name_str	Defines the name of the IP pool.
ip_addr1	Sets the starting IP address in the IP pool.
ip_addr2	Sets the ending IP address in the IP pool.

Example: To configure the IP pool named “office” with the IP addresses 172.16.10.100 through 172.16.10.200:

set ippool office 172.16.10.100 172.16.10.200

l2tp

Use the l2tp commands to configure or remove L2TP (Layer 2 Tunneling Protocol) tunnels and L2TP settings from the security device.

L2TP is an extension to PPP (Point-to-Point Protocol) that allows ISPs to operate VPNs. L2TP allows dial-up users to make virtual Point-to-Point Protocol (PPP) connections to an L2TP network server (LNS). The security device can operate as such a server.

Syntax

clear

```
clear [ cluster ] l2tp { all | ip ip_addr }
```

get

```
get l2tp
    {
    all [ active ] | tunn_str [ active ] |
    default
    }
```

set (default)

```
set l2tp default
    {
    auth server name_str [ query-config ] |
    ippool string |
    dns1 ip_addr | dns2 ip_addr |
    wins1 ip_addr | wins2 ip_addr |
    ppp-auth { any | chap | pap } |
    }
```

set (tunn_str)

```
    set l2tp tunn_str
      [
      auth server name_str
        [ query-config ] [ user usr_name | user-group grp_name ] |
      [ peer-ip ip_addr ]
        [ host name_str ]
          [ outgoing-interface interface ]
            [ secret string ]
              [ keepalive number ] |
      remote-setting
        { [ ippool string ]
          [ dns1 ip_addr ]
            [ dns2 ip_addr ]
              [ wins1 ip_addr ]
                [ wins2 ip_addr ]
        }
      ]
```

Keywords and Variables

Variables

```
get l2tp tunn_str [ ... ]
set l2tp tunn_str [ ... ]
unset l2tp tunn_str { ... }
```

tunn_str The name or IP address of the L2TP tunnel.

Example: The following command identifies the RADIUS authentication server (Rad_Serv) for an L2TP tunnel (Mkt_Tun).

set l2tp Mkt_Tun auth server Rad_Serv

active

get l2tp all active
get l2tp *tunn_str* active

active Displays the currently active L2TP connections for tunnels.

Example: The following command displays the current active/inactive status of the L2TP connection for a tunnel (home2work):

get l2tp home2work active

all

clear cluster l2tp all
clear l2tp all
get l2tp all

all Displays or clears the ID number, tunnel name, user, peer IP address, peer hostname, L2TP tunnel shared secret, and keepalive value for every L2TP tunnel (all) or a specified L2TP tunnel (*string*).

auth server

set l2tp *tunn_str* auth server *name_str* [...]
set l2tp default auth server *name_str* [...]
unset l2tp *tunn_str* auth

auth server Specifies the object name (*name_str*) of the authentication server containing the authentication database.

- query-config Directs the security device to query the authentication server for IP, DNS, and WINS information.
- user *usr_name* Restricts the L2TP tunnel to a specified user (*usr_name*).
- user-group *grp_name* Restricts the L2TP tunnel to a specified user group (*grp_name*).

Example: The following command directs the device to query the RADIUS authentication server (Rad_Serv) for IP, DNS, and WINS information:

set l2tp Mkt_Tun auth server Rad_Serv query-config

cluster

clear cluster l2tp { ... }

cluster Propagates the clear operation to all other devices in an NSRP cluster.

default

```
get l2tp default
set l2tp default { ... }
unset l2tp tunn_str [ ... ]
unset l2tp default { ... }
```

default

Defines or displays the default L2TP settings.

- auth server *name_str* The object name of the authentication server.
- dns1 *ip_addr* The IP address of the primary DNS server.
- dns2 *ip_addr* The IP address of the secondary DNS server.
- ippool *string* The name of the L2TP IP pool, from which IP addresses are drawn to be assigned to L2TP users.
- ppp-auth { any [chap | pap] } Specifies the authentication type in response to a dialup user's request to make a Point-to-Point Protocol (PPP) link. (The any switch instructs the security device to negotiate CHAP and then, if that attempt fails, PAP.)
  - chap specifies Challenge Handshake Authentication Protocol (CHAP), which does not transmit the password across the network.
  - pap specifies Password Authentication Protocol (PAP), which does not use encryption.
- radius-port *port_num* Defines the port number of the default L2TP server. The number can be between 1024 and 65,535.
- wins1 *ip_addr* The IP address of the primary WINS server.
- wins2 *ip_addr* The IP address of the secondary WINS server.

Example: The following commands create a set of default L2TP settings.

- IP pool (chiba).
- Use of the local database.
- CHAP for PPP authentication.
- Primary and secondary DNS servers at 192.168.2.1 and 192.168.4.71 respectively.
- Primary and secondary WINS servers at 10.20.1.16 and 10.20.5.101 respectively.

```
set l2tp default ippool chiba
set l2tp default auth local
set l2tp default ppp-auth chap
set l2tp default dns1 192.168.2.1
set l2tp default dns2 192.168.4.71
set l2tp default wins1 10.20.1.16
set l2tp default wins2 10.20.5.101
```

host

set l2tp *tunn_str* […] host *name_str* […]
unset l2tp *tunn_str* host

host	Adds a restriction that allows only a client with the specified client hostname (*name_str*) to establish the L2TP tunnel.

keepalive

set l2tp *tunn_str* […] keepalive *number*

keepalive	Defines how many seconds of inactivity, the security device (LNS) waits before sending a hello message to the dialup client (LAC).

Example: The following command specifies a keepalive value of 120 for an L2TP tunnel (west_coast):

set l2tp west_coast keepalive 120

outgoing-interface

set l2tp *tunn_str* […] outgoing-interface *interface*

outgoing-interface	Specifies the outgoing interface for the L2TP tunnel. Note: This setting may be mandatory on your security device.

Example: The following command specifies interface ethernet4 as the outgoing interface for L2TP tunnel (east_coast):

set l2tp east_coast outgoing-interface ethernet4

peer-ip

set l2tp *tunn_str* […] peer-ip *ip_addr* […]

peer-ip	Adds a restriction that allows only a client host with the specified IP address (*ip_addr*) to establish the L2TP tunnel.

Example: The following command specifies the IP address of the LAC (172.16.100.19):

set l2tp east_coast peer-ip 172.16.100.19

secret

set l2tp *tunn_str* […] secret *string* […]

secret	Defines a shared secret used for authentication between the security device (which acts as the L2TP Network Server, or LNS) and the L2TP access concentrator (LAC).

Example: The following command specifies a shared secret (94j9387):

set l2tp east_coast secret 94j9387

user

set l2tp *tunn_str* auth server *name_str* [...] user *usr_name*

user	Restricts the L2TP tunnel to a L2TP user (*usr_name*). Not specifying *name_str* enables any L2TP user.

Example: The following command adds a restriction that allows only a specified L2TP user (jking) to establish a L2TP tunnel (west_coast).

set l2tp west_coast auth server Our_Auth user jking

Defaults

The default L2TP UDP port number is 1701.

By default, the security device uses no L2TP tunnel secret to authenticate the LAC-LNS pair. This is not a problem, because the device performs IKE authentication when it uses L2TP over IPSec.

The default interval for sending a keepalive message is 60 seconds.

PPP-auth type is **any**.

license-key

Use the **license-key** command to upgrade or display the current software license.

The license key feature allows you to expand the capabilities of your security device without having to upgrade to a different device or system image. You can purchase a key that unlocks specified features already loaded in the software, such as the following:

- User capacity
- Virtual systems
- Zones
- Virtual routers
- HA

Syntax

exec

```
exec license-key option_string
    {
    delete key_str |
    nsrp key_str |
    update option_string |
    vrouter key_str |
    vsys key_str |
    zone key_str
    }
```

get

```
get license-key
```

set

```
set license-key update-url url_str
```

Keywords

delete

exec license-key delete *key_str*

delete	Deletes the license key (*key_str*).

nsrp

exec license-key nsrp *key_str*

nsrp	Specifies a NetScreen Redundancy Protocol (NSRP) license key (*key_str*).

update

exec license-key update

update	Before your security device can receive regular update service for Deep Inspection (DI) signatures, you must purchase a subscription to the service, register your device, and then retrieve the subscription. You retrieve the subscription and activate it on your device by executing the command **exec license-key update**. For more information, refer to the *Concepts and Examples ScreenOS Reference Guide.*

update-url

set license-key update-url *url_str*

update-url	Specifies the URL of the license key server from which the security device loads license key updates.

vrouter

exec license-key vrouter *key_str*

vrouter	Specifies a virtual router license key (*key_str*).

vsys

exec license-key vsys *key_str*

vsys	Specifies a virtual system (vsys) license key (*key_str*).

zone

exec license-key zone *key_str*

zone	Specifies a security zone license key (*key_str*).

log

Use the log commands to configure the security device for message logging.

The the log commands allow you to perform the following tasks:

- Display the current log status according to severity level, policy, service, ScreenOS module, source, destination, or duration
- Determine which log information to display or omit
- Display asset recovery information
- Configure logging to mitigate message loss due to memory limitations

Syntax

clear

```
clear [ cluster ] log
    {
    self [ end-time string ] |
    system [ saved ] |
    traffic [ policy id_num [ -id_num ] [ end-time string ] ]
    }
```

get

```
get log
    {
    asset-recovery |
    audit-loss-mitigation |
    self | traffic [ policy pol_num [ -pol_num ] ]
      [ start-date date [ time ] ] [ end-date date ] [ time ] ]
        [ start-time string ] [ end-time string ]
          [ min-duration string ] [ max-duration string ]
            [ service name_str ]
              [ src-ip ip_addr [ -ip_addr ]
                [ src-netmask mask ] [ src-port port_num ]
            ]
                  [ dst-ip ip_addr [ -ip_addr ] [ dst-netmask mask ] ]
                    [ dst-port port_num ]
                      [ no-rule-displayed ] |
      sort-by
        {
        date [ [ start-date date [ time ] ] [ end-date date [ time ] ] ]
        dst-ip [ ip_addr [ -ip_addr | dst-netmask mask ] ]
        src-ip [ ip_addr [ -ip_addr | src-netmask mask ] ]
        time
          [ start-time time ]
            [ end-time time ]
        }
    setting [ module { system | all } ]
    }
```

set

```
set log
    {
    audit-loss-mitigation |
    cli { enable | file-size number } |
    module name_str level string destination string
    }
```

Keywords and Variables

audit-loss-mitigation

```
get log audit-loss-mitigation
set log audit-loss-mitigation
unset log audit-loss-mitigation
```

audit-loss-mitigation

Stops generation of auditable events when the number of such events exceeds the capacity of the security device. Enabling this feature reduces the loss of event logs due to log overloads.

On some security devices, you must connect the syslog server to the management interface on the Management Module. This ensures that the syslog server is available if the audit trail fills up and network traffic stops.

cli

set log cli { enable **|** file-size *number* }

cli	Enables logging CLI activity. You can specify a limit in bytes for the size of the log.

cluster

clear cluster log { ... }

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

destination

set log module *name_str* level *string* destination *string*
unset log module *name_str* level *string* destination *string*

destination	Specifies the destination of the generated log messages. The permissible destinations are console, internal, email, snmp, syslog, webtrends, onesecure, and pcmcia.

Example: The following command instructs the security device to direct all system module messages at the alert level (or higher) to the console port.

set log module system level alert destination console

dst-port

get log traffic dst-port *number*

dst-port	Filters the output of the get log command by a range of destination port numbers or by a specific destination port number.

Example: The following command filters the get traffic log output to display only traffic destined for port 80 (that is, HTTP traffic):

get log traffic dst-port 80

level

set log module *name_str* level *string* destination *string*
unset log module *name_str* level *string* destination *string*

level	Specifies the minimum urgency level of the generated log messages. Starting with the most urgent, these levels are emergency, alert, critical, error, warning, notification, information, and debugging. For the get log command, the all-levels option displays all security levels.

Example: The following command instructs the security device to direct all system module messages at the critical level (or higher) to the email server:

set log module system level critical destination email

min-duration | max-duration

get log event { … } […] min-duration *string* […]
get log event { … } […] max-duration *string* […]

min-duration Displays traffic log entries for traffic whose duration was longer than or equal to the minimum duration specified.

max-duration Displays traffic log entries for traffic whose duration was shorter than or equal to the maximum duration specified.

Example: The following command displays traffic log entries for traffic that lasted 5 minutes to 1 hour:

get log traffic min-duration 00:05:00 max-duration 01:00:00

module

get log event module { … } […]
set log module *name_str* { … }
unset log module *name_str* { … }

module Specifies the name of the ScreenOS module that generates the log message.

Example: The following command instructs the security device to direct all system module messages at the critical level (or higher) to the webtrends server:

set log module system level critical destination webtrends

no-rule-displayed

get log { … } […] no-rule-displayed

no-rule-displayed Displays traffic log entries, but does not display policy information.

Example: The following command displays traffic log entries without displaying policy information:

get log traffic no-rule-displayed

policy

clear [cluster] log traffic policy *pol_num* […]

policy Displays traffic log entries for a policy (specified by its ID number) or for several policies (specified by a range of ID numbers). The ID number can be any value between 0 and the total number of established policies. To define a range, enter the starting and ending ID numbers using this syntax: *pol_num - pol_num*

Example: The following command displays traffic log table entries for any policy with ID 3 to 9 (inclusive):

get log traffic policy 3-9

self

clear [cluster] log self [...]
get log self [...]

self	Clears or displays self-log entries from the log.

Example: The following command displays traffic log table entries for any policy with a source IP address of 172.16.10.1 and a destination address of 172.16.10.100:

get log self src-ip 172.16.10.1 dst-ip 172.16.10.100

service

get log { ... } [...] service *name_str* [...]

service	Displays traffic log entries for a specified service, such as TCP, ICMP, FTP, or Any. The name does not have to be complete; for example, both TC and CP are recognized as TCP. Although you cannot specify a service group, TP is recognized as FTP, HTTP, and TFTP, entering TP displays log entries for all three services.

Example: The following command displays traffic log table entries for TCP:

get log self service tcp

setting

get log setting [...]

setting	Displays log setting information. The module *string* value specifies the name of the module for which the log settings apply.

Example: The following command displays traffic log settings for the system module:

get log setting module system

sort-by

get log { ... } sort-by date [[start-date *date*] [end-date *date*]] [time]
get log { ... } sort-by dst-ip [*ip_addr* [*-ip_addr* | dst-netmask *mask*]]
get log { ... } sort-by src-ip [ip_addr [*-ip_addr* | src-netmask *mask*]]
get log { ... } sort-by time [start-time *time*] [end-time *time*]

sort-by

Sorts the log information by any of the following criterion.

- date | time Sorts the logs by date, time, or both.

  The start-date option displays logs that occurred at or before the time specified. The end-date option displays logs that occurred at or after the time specified. The format for start-date and end-date *date* is *mm/dd*[*yy-hh:mm:ss*]. The format for start-time and end-time is *hh:mm:ss*.

  You can omit the year (the current year is the default), or express the year using the last two digits or all four digits. The hour, minute, and second are optional. The delimiter between the date and the time can be a dash or an underscore:

  12/31/2002-23:59:00

  12/31/2002_23:59:00

- dst-ip Sorts the traffic logs by destination IP address.
- src-ip Sorts the traffic logs by source IP address.

Example: The following command displays traffic log settings sorted by date and time:

get log traffic sort-by date start-date 11/21/2003-22:24:00

src-ip | dst-ip

get log { ... } [...] src-ip *ip_addr* [*-ip_addr*] [...]
get log { ... } [...] sort-by src-ip *ip_addr* [*-ip_addr*] [...]
get log { ... } [...] dst-ip *ip_addr* [*-ip_addr*] [...]
get log { ... } [...] sort-by dst-ip *ip_addr* [*-ip_addr*] [...]

src-ip

Displays traffic log entries for a specified source IP address or range of source IP addresses. Include the subnet mask for a source IP address to display traffic entries for all IP addresses in the same subnet as the specified source IP address. You cannot specify a source IP range and a source subnet mask simultaneously. You can also direct the device to sort event logs by source IP address.

dst-ip

Displays traffic log entries for a specified destination IP address or range of destination IP addresses. You can specify the subnet mask for a destination IP address, but you cannot specify a destination IP range and destination subnet mask simultaneously. You can also direct the device to sort event logs by destination IP address.

Example: The following command displays traffic log entries for the range of destination IP addresses 172.16.20.5–172.16.20.200:

get log traffic dst-ip 172.16.20.5-172.16.20.200

src-port

get log { ... } [...] src-port *port_num* [...]

src-port

Displays traffic log entries for a specified port number or range of source port numbers.

Example: The following command displays traffic log entries from the source port 8081:

get log traffic src-port 8081

start-date | end-date

get log { ... } [start-date *date* [*time*]] [end-date *date* [*time*]]
get log { ... } sort-by date [start-date *date* [*time*]] [end-date *date* [*time*]]

start-date *date [time]*
end-date *date [time]*

Specifies the lower and upper ends of a range of dates for traffic or self logs. You can omit the year (the current year is the default), or express the year using the last two digits or all four digits. The hour, minute, and second are optional. The delimiter between the date and the time can be a dash or an underscore:

12/31/2001-23:59:00

12/31/2001_23:59:00

start-time | end-time

get log { ... } [start-time *time*] [end-time *time*]
get log { ... } sort-by [start-time *time*] [end-time *time*]

start-time
end-time

Specifies the lower and upper ends of a range of times for traffic or self logs. When you specify a start-time and/or end-time, the device sorts or filters the logs based on the specified times, regardless of the date. Specify the time in the following format: hh:mm:ss

Example: The following command displays event log entries from 3:00 P.M. on March 4, 2001 to 2:59:59 P.M. on March 6:

get log event start-time 03/04/01_15:00 end-time 03/06_14:59:59

system

clear [cluster] log system [...]
get log system [reversely | saved]

system

Displays current system log information. The saved switch displays saved system log information. The reversely switch displays information in reverse order.

Example: The following command generates log messages generated from module system, and to generate only messages that are critical or greater:

set log module system level critical destination console

traffic

```
clear [ cluster ] log traffic [ ... ]
get log traffic [ ... ]
```

traffic	Specifies traffic log entries.

Example: The following command displays traffic log entries from the source port 8081:

get log traffic src-port 8081

mac

Use the mac commands to configure a static Media Access Control (MAC) address for a Juniper Networks interface or to display information about the current MAC configurations.

NOTE: You can only execute the mac commands when the device is configured in Transparent mode.

Syntax

set

set mac *mac_addr interface*

get

get mac *interface*

Keywords and Variables

Variables

mac_addr	Specifies the MAC address.
interface	Specifies the name of the interface, as with ethernet1.

Example: The following command sets the MAC address on an security device to 111144446666 for the ethernet7 interface:

set mac 111144446666 ethernet7

mac-learn

Use the **mac-learn** commands to clear the entries in the Media Access Control (MAC) learning table, or to display information about the current MAC configurations.

This command functions only when an interface is in Transparent mode. When interfaces are in Transparent mode, the security device operates at Layer 2. The security zone interfaces do not have IP addresses, and the security device forwards traffic like a Layer 2 switch.

Syntax

clear

clear [cluster] mac-learn [stats]

get

get mac-learn [*interface*]

Keywords and Variables

Variables

get mac-learn *interface*

interface	Identifies the interface.

cluster

clear cluster mac-learn [...]

cluster	Propagates the **clear** operation to all other devices in an NSRP cluster.

stats

clear [cluster] mac-learn stats

stats	Clears the MAC learning table statistics.

memory

Use the **memory** commands to set or display the memory allocation conditions.

Syntax

get memory [*id_num* **|** all **|** cache **|** error **|** free **|** module { all **|** *id_num* } **|** used]

Keywords and Variables

Variables

get memory *id_num*

id_num	The task ID number.

all

get memory all

all	Displays memory fragments.

cache

get memory cache

cache	Displays malloc cache.

error

get memory error

error	Displays erroneous memory fragments.

free

get memory free

free	Displays free memory.

ipc

get memory ipc

ipc	Displays memory statistics about inter-process communication (IPC).

kernel

get memory kernel [...]

kernel	Displays memory statistics about the kernel heap.

module

get memory module *id_num*

module	Displays a single memory module (*id_num*).

pool

```
get memory pool
```

pool	Displays pooled memory.

used

```
get memory used
```

used	Displays used memory.

mirror

Use the **mirror** commands to mirror all traffic for at least one source interface to a destination interface. This command is useful for debugging and monitoring network traffic. For example, you can connect a sniffer to a destination interface to monitor traffic passing through multiple source interfaces.

NOTE: When a destination interface mirrors multiple source interfaces, the device may drop some frames due to a bandwidth mismatch.

Syntax

get

```
get mirror port
```

set

set mirror port source *interface1* destination *interface2*

Keywords and Variables

destination | source

set mirror port source *interface1* destination *interface2*

destination	Specifies the source and destination interfaces.

ndp

Use the ndp command to list neighbor discovery (ND) entries in the neighbor cache table or create an entry in the table.

ND is the process of tracking the reachability status for neighbors in a local link. A security device views a neighbor as reachable when the device receives recent confirmation that the neighbor received and processed IP traffic or Neighbor Solicitation (NS) requests. Otherwise, it considers the neighbor unreachable.

Syntax

clear

```
clear [ cluster ] ndp
```

get

```
get ndp
```

set

```
set ndp
    {
    ip_addr mac_addr interface |
    always-on-dest
    }
```

Keywords and Variables

Variables

set ndp *ip_addr mac_addr interface*

ip_addr	The IPv6 address for the interface in the ND (Neighbor Discovery) entry in the neighbor cache table.
mac_addr	The MAC address for the interface in the neighbor cache table entry.
interface	The name of the ND interface. For more information, see “Interfaces” on page 319.

always-on-dest

```
set ndp always-on-dest
unset ndp always-on-dest
```

always-on-dest	Directs the security device to send an ND request for any incoming packet with a heading containing a MAC address not yet listed in the device’s MAC address table. This may be necessary when packets originate from server load-balancing (SLB) switches or from devices using the Hot Standby Router Protocol/Virtual Router Redundancy Protocol (HSRP/VRRP).

cluster

clear [**cluster**] **ndp**

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

nsmgmt

Use the nsmgmt commands to set up a security device for configuration and monitoring by NetScreen-Security Manager (NSM) 2004, an enterprise-level management application that configures multiple security devices from remote hosts.

The nsmgmt command can modify settings for two NSM components.

- The *management system*, a set of services that reside an external host (called a *device server*). These services process, track, and store device management information.

- The *Agent*, a service that resides on the managed security device. The Agent receives configuration parameters from the management system and pushes it to ScreenOS. The Agent also monitors the device and transmits reports back to the management system.

NOTE: We do not recommend using the nsmgmt command to initiate communication between NetScreen-Security Manager and the NSM Agent. The method for performing this initiation differs depending upon the current availability of the security device.

If your security device has a known, accessible IP address, you can set initial configuration parameters from a client through the NSM UI. You can also use the UI if your device receives IP addresses dynamically, as with DHCP or PPPoE. If the device is new and does not have a known, accessible IP address, you can perform simple initial configuration by executing an encrypted configuration file call Configlet, which automatically executes several nsmgmt commands. Your network security administrator creates the Configlet file on the NSM UI and delivers it to your remote site, usually manually or by email.

For more information, refer to the discussion about adding devices in the *NetScreen-Security Manager 2004 Administrator's Guide.*

Syntax

get

```
get nsmgmt
    [
    proto-dist
      {
      table { bytes | packets } |
      user-service
      }
    ]
```

set

```
set nsmgmt
    {
    enable |
    init
      {
      id string |
      installer name name_str password pswd_str |
      otp string
      } |
    report
      {
      alarm { attack | di | other | traffic } enable |
      log { config | info | self | traffic } enable |
      proto-dist
        {
        enable |
        user-service svc_name { ah | esp | gre | icmp | ospf | tcp | udp }
          { port_num1-port_num2 }
        } |
      statistics { attack | ethernet | flow | policy } enable
      }
    server
      primary | secondary
        { name_str | ip_addr } [ port number ] [ src-interface interface ] |
    }
```

Keywords and Variables

enable

```
get nsmgmt enable
set nsmgmt enable
```

enable — Enables remote management by initiating contact with the management server.

init

```
get nsmgmt init
set nsmgmt init id string
set nsmgmt init installer name name_str password pswd_str
set nsmgmt init otp string
```

init — Sets initialization parameters for interaction with the management server.

- id *string* An ID used (only once) during initiation of the connection between the security device and the management server. The security device passes the ID to the Management System to look up the One-Time Password in the management database.
- installer name *name_str* password *pswd_str* Specifies an installer name and password, used (only once) during initiation of the connection between the security device and the management server.
- otp *string* Sets the One-Time Password (OTP). The security device uses this password one time to contact the Security Management system. After initiation of contact between the device and the management database, the device executes an unset command to erase the OTP.

keys

```
get nsmgmt keys
```

keys — Displays information about the public and private keys used to encrypt and decrypt the Configlet file.

report

```
set nsmgmt report { alarm | log | proto-dist | statistics } { ... }
unset nsmgmt report { alarm | log | proto-dist | statistics } { ... }
```

report — Specifies which event messages the security device transmits to the server.

alarm Enables the transmission of alarm events. The categories of alarms are as follows.

- attack Transmits attack alarms such as syn-flag or syn-flood. For more information about such attacks, see “interface” on page 118.
- di Transmits attack alarms generated during Deep Inspection.
- traffic Transmits traffic alarms.
- other Transmits alarms other than attack, Deep Inspection, or traffic alarms.

The enable switch enables messaging for the specified alarm message.

log Enables the transmission of log events. The categories of logs are as follows.

- **config** Transmits log messages for events triggered by changes in device configuration.
- **info** Transmits low-level notification log messages about noncritical changes that occur on the device, as when an authentication procedure fails.
- **self** Transmits log messages concerning dropped packets (such as those denied by a policy) and traffic that terminates at the security device (such as administrative traffic). The self log displays the date, time, source address/port, destination address/port, duration, and service for each dropped packet or session terminating at the security device.

proto-dist Sets or displays parameters for transmission of messages concerning protocol distribution parameters.

- **enable** Enables transmission of protocol distribution messages to the server.
- **table** Displays the number of bytes or packets transmitted to the protocol distribution table.
- **user-service** *svc_name* Specifies messages generated by the following services.
  - **ah** AH (Authentication Header) service.
  - **esp** ESP (Encapsulating Security Payload) service.
  - **gre** GRE (Generic Routing Encapsulation).
  - **icmp** ICMP (Internet Control Message Protocol).
  - **ospf** OSPF (Open Shortest Path First).
  - **tcp** TCP (Transmission Control Protocol).
  - **udp** UDP (User Datagram Protocol).

The *port_num1-port_num2* setting specifies a range of port numbers.

- **traffic** Transmits alarms generated while the device monitors and records the traffic permitted by policies. A traffic log notes the following elements for each session:
  - Date and time that the connection started
  - Source address and port number
  - Translated source address and port number
  - Destination address and port number
  - The duration of the session
  - The service used in the session

The **enable** switch enables messaging for the specified log message.

statistics Enables the security device for reporting statistical information to the server.

- **attack** Enables transmission of messages containing attack statistics.
- **ethernet** Enables transmission of messages containing ethernet statistics.
- **flow** Enables transmission of messages containing traffic flow statistics.
- **policy** Enables transmission of messages containing policy statistics.

The **enable** switch enables messaging for the specified statistical message.

server

```
get nsmgmt server
set nsmgmt server { primary | secondary } { name_str | ip_addr } [ port number ]
```

server	Identifies the Security Management system server.

Chapter 3
ntp Through RIPng

This section lists and describes the ScreenOS command line interface (CLI) commands **ntp** through **RIPng**.

NOTE: Certain commands and features are platform-dependent and might be unavailable on your Juniper Networks security product platform. Check your product datasheet for feature availability.

Click on a command for details:

ntp
os
override
performance
ping
pki
policy
port-mode
pppoe
proxy-id
reset
RIPng Commands

ntp

Use the **ntp** commands to configure the security device for Simple Network Time Protocol (SNTP).

SNTP is a simplified version of NTP, which is a protocol used for synchronizing computer clocks in the Internet. This version is adequate for devices that do not require a high level of synchronization and accuracy. To enable the SNTP feature, use the **set clock ntp** command.

Syntax

exec

```
exec ntp [ server { backup1 | backup2 | primary } ] update
```

get

```
get ntp
```

set

```
set ntp
    {
    auth { preferred | required } |
    interval number |
    max-adjustment number |
    no-ha-sync |
    server [ { backup1 | backup2 } { ip_addr | dom_name } ]
       key-id number preshare-key string |
    timezone number1 number2
    }
```

Keywords and Variables

auth

```
set ntp auth { preferred | required }
```

auth	Configures an authentication mode to secure NTP traffic between the security device and the NTP server. ■ **required** The Required mode specifies that the security device must authenticate all NTP packets using the key id and preshared key information that the security device and the NTP server previously exchanged out of band (the device does not exchange the preshared key over the network). ■ **preferred** The Preferred mode specifies that the security device first must try to authenticate all NTP packets by sending out an update request that includes authentication information—key id and checksum—same as for the Required mode. If authentication fails, the security device then sends out another update request without the authentication information. Note: Before you can set an authentication mode, you must assign a key id and preshared key to at least one of the NTP servers configured on the security device.

interval

```
set ntp interval number
unset ntp interval
```

interval	Defines in minutes how often the security device updates its clock time by synchronizing with the NTP server. The range for the synchronization interval is from 1 to 1440 minutes (24 hours).

Example: The following command configures the security device to synchronize its clock time every 20 minutes:

```
set ntp interval 20
```

max-adjustment

```
set ntp max-adjustment number
unset ntp max-adjustment
```

max-adjustment

Configures a maximum time adjustment value. This value represents the maximum acceptable time difference between the security device system clock and the time received from an NTP server. When receiving a reply from an NTP server, the security device calculates the time difference between its system clock and the NTP server and updates its clock only if the time difference between the two is within the maximum time adjustment value that you set.

no-ha-sync

```
set ntp no-ha-sync
unset ntp no-ha-sync
```

no-ha-sync

In a High Availability (HA) configuration, instructs the security device not to synchronize its peer device with the NTP time update.

server

```
set ntp server { ip_addr | dom_name }
set ntp server key-id number preshare-key string
set ntp server { backup1 | backup2 } { ip_addr | dom_name }
set ntp server { backup1 | backup2 } key-id number preshare-key string
unset ntp server { ... }
```

server

- *ip_addr* The IPv4 or IPv6 address of the primary NTP server with which the security device can synchronize its system clock time.
- *dom_name* The domain name of the primary NTP server with which the security device can synchronize its system clock time.
- backup1 | backup2
  - *ip_addr* The IPv4 or IPv6 address of the first (or second) backup NTP server with which the security device can synchronize its system clock time in case the primary server is not available.
  - *dom_name* The domain name of the first (or second) backup NTP server with which the security device can synchronize its system clock time in case the primary server is not available.
  - key-id *number* Assigns a key id to the current server for authentication purposes
  - preshare key Assigns a preshared key to the current server for authentication purposes.
- key-id *number* Assigns a key id to the current server for authentication purposes.
- preshare key Assigns a preshared key to the current server for authentication purposes.

timezone

set ntp timezone *number1 number2*
unset ntp timezone

timezone	Defines the Time Zone, expressed as an integer *number1* between -12 and 12 inclusive. A value of zero denotes GMT (Greenwich Mean Time). *number2* expresses minutes.

Example: The following command sets the Time Zone to Greenwich Mean time:

set ntp timezone 0

update

exec ntp update

update	Updates the time setting on a security device to synchronize it with the time setting on an NTP server.

os

Use the os commands to display kernel and task information for the operating system of the security device.

Syntax

get os { cost **|** flow **|** kernel **|** misc **|** task *name_str* }

Keywords and Variables

cost

get os cost

cost	Displays the amount of processor time used by elements of the operating system.

flow

get os flow

flow	Displays flow statistics.

kernel

get os kernel

kernel	Displays kernel statistics.

misc

```
get os misc
```

misc	Displays miscellaneous information.

task

```
get os task name_str
```

task	Displays information about a specified task (*name_str*).

override

Use the **override** commands to override the following vsys parameters (which are defined using the **vsys-profile** commands):

- CPU weight
- Sessions (maximum and reserved values and alarm threshold)

The override commands are only available after you enter a vsys. By default, no override values exist.

Syntax

get

```
get override [ cpu-weight | session-limit ]
```

set

```
set override
    {
    cpu-weight number |
    session-limit { alarm number | max number | reserve number }
    }
```

Keywords and Variables

cpu-weight

```
get override [ cpu-weight ]
set override cpu-weight number
unset override cpu-weight
```

cpu-weight	CPU weight for the vsys. After entering the vsys, you can set an override value for the CPU weight defined in the vsys profile. Use the **unset override cpu-weight** command to remove the override. The CPU weight configured in the vsys profile is now used.

Example: The following commands first enter the vsys named hr and then override the CPU weight to 30.

```
device-> enter vsys hr
device(hr)-> set override cpu-weight 30
device(hr)->
```

session-limit

```
get override [ session-limit ]
set override session-limit { alarm number | max number | reserve number }
unset override session-limit { alarm | max | reserve }
```

session-limit

Specifies session-limit override for the vsys:

- **alarm:** Specifies the percentage of the session limit at which an alarm is triggered. The alarm value is from 1 through 100 percent.
- **max:** Maximum number of sessions for the vsys. The configured maximum session value cannot exceed the absolute maximum value for the security device.
- **reserve:** Number of reserved sessions for the vsys when the security device becomes oversubscribed. The reserved session value cannot exceed the maximum session value.

Use the **unset override session-limit** command to remove the override. The session-limit values configured in the vsys profile are now used.

Example: The following commands first enter the vsys named hr and then override the maximum number of sessions to 4000.

```
device-> enter vsys hr
device(hr)-> set override session-limit max 4000
device(hr)->
```

performance

Use the **performance** commands to retrieve performance information for the security device.

You can display information for the following:

- CPU utilization
- The session ramp-up rate

Syntax

```
get performance
    {
    cpu [ detail ] |
    cpu-limit [ detail [ vsys { vsys | all } ] ] |
    session [ detail ]
    }
```

Keywords and Variables

cpu

```
get performance cpu [ detail ]
```

cpu-limit

```
get performance cpu-limit [ detail [ vsys { name | all } ] ]
```

cpu-limit

If the CPU limit feature is enabled, displays the CPU weights and configured CPU quota percentage for all virtual systems. Also displays percentage of CPU quota used for the last minute, the last 5 minutes, and the last 15 minutes.

- **all** displays detailed CPU limit performance information for all virtual systems.
- **detail** displays CPU limit performance information for the last 60 seconds, the last 60 minutes, and the last 24 hours.
- **vsys** displays detailed CPU limit performance information for the specified vsys.

session

```
get performance session [ detail ]
```

session

Displays the number of sessions added (ramp-up rate) for the last minute, the last 5 minutes, and the last 15 minutes. It does not display the total number of sessions or the number of deleted sessions.

Displays the number of sessions added (ramp-up rate) for the last minute, the last 5 minutes, and the last 15 minutes. It does not display the total number of sessions or the number of deleted sessions.

- **detail** displays session ramp-up rate for the last 60 seconds, the last 60 minutes, and the last 24 hours.

ping

Use the **ping** commands to check the network connection to another system.

NOTE: An extended **ping** (using the **from** option) pings a host on the Untrusted network from any existing MIP or from the Trusted interface IP address. The syntax for specifying a MIP is **mip** *ip_addr* (see example in **from** keyword description).

Syntax

```
ping [ ip_addr | name_str ]
    [ count number [ size number [ time-out number ] ] ]
      [ from interface ]
    name-lookup [ outgoing-interface ]
```

Keywords and Variables

Variables

ping [*ip_addr* **|** *name_str*] […]

ip_addr **|** *name_str*	Pings the host at an IPv4 or IPv6 address (*ip_addr*) or with name (*name_str*).

Example 1: The following command pings a host with IP address 172.16.11.2:

ping 172.16.11.2

Example 2: The following command pings a host with IPv6 address 3::200:ff:fe00:6:

ping 3::200:ff:fe00:6

count

ping [*ip_addr* **|** *name_str*] count *number* […]

count	The ping count (*number*).

Example: The following command pings a device at 10.100.2.171 with a ping count of 3:

ping 10.100.2.171 count 3

from

ping [*ip_addr* **|** *name_str*] […] from *interface*

from	The source interface (*interface*) for an extended ping. For more information, see “Interfaces” on page 319. Defines the source IP to which the ping will reply. Because this destination is on the untrusted side, the source IP can only be the Mapped IP address or an untrusted interface IP address.

Examples: The following command pings a device at 10.100.2.11 with a ping count of 4 from the ethernet1 interface:

ping 10.100.2.11 count 4 from ethernet1

The following command pings a host with IP address 192.168.11.2 and sends the results to IP address 10.1.1.3:

ping 192.168.11.2 from mip 10.1.1.3

name-lookup

ping *ip_addr* name-lookup [outgoing-interface]

name-lookup — Uses the ICMP name to do a name lookup instead of using an echo request.

outgoing-interface automatically selects the outgoing interface to do the lookup.

size

ping [*ip_addr* **|** *name_str*] count *number* size *number* […]

size — The packet size (*number*) for each ping.

time-out

ping [*ip_addr* **|** *name_str*] count *number* size *number* time-out *number*

time-out — The ping timeout in seconds (*number*).

Example: The following command pings a device at 10.100.2.11:

- Ping count of 4
- Packet size 1000
- Ping timeout of 3 seconds

ping 10.100.2.11 count 4 size 1000 time-out 3

pki

Use the **pki** commands to manage public-key infrastructure (PKI).

PKI refers to the hierarchical structure of trust required for public key cryptography. Using PKI, the security device verifies the trustworthiness of a certificate by tracking a path of certificate authorities (CAs) from the one issuing your local certificate back to a root authority of a CA domain.

The **pki** commands perform the following tasks:

- Manage PKI objects.
- Create new RSA key pairs and acquire a certificate.
- Verify the certificate received from the communication peer.
- Acquire CRLs.
- Configure PKI-related operations, such as verification of certificate revocation.

Syntax

exec

```
exec pki
    {
    convert-cert |
    dsa new-key number [ & ] |
    rsa new-key number [ & ] |
    x509
      {
      install-factory-certs name_str |
      pkcs10 |
      scep
        {
        cert id_num |
        key { id_num | last-key } |
        renew id_num |
      } |
      tftp ip_addr { cert-name name_str | crl-name name_str }
      }
    }
```

get

```
get pki
    {
    authority { id_num | default }
      {
      cert-path |
      cert-status |
      scep
      } |
    ldap |
    pre-prime |
    src-interface |
    x509
      {
      cert id_num |
      cert-fqdn |
      cert-path |
      crl-refresh |
      dn |
      list { ca-cert | cert | crl | key-pair | local-cert | pending-cert } |
      pkcs10 |
      raw-cn |
      send-to
      }
    }
```

set (authority)

```
set pki authority { id_num | default }
    {
    cert-path { full | partial } |
    cert-status
      {
      crl
        {
        refresh { daily | default | monthly | weekly } |
        server-name { ip_addr | dom_name } |
        url url_str
        }
      ocsp
        {
        cert-verify id id_num |
        not-verify-revoke |
        url url_str
        } |
      revocation-check { crl [ best-effort ] | ocsp [ best-effort ] | none }
      }
    scep
      {
      authentication { failed | passed } |
      ca-cgi string |
      ca-id name_str |
      challenge pswd_str |
      current |
      mode { auto | manual } |
      polling-int number |
      ra-cgi string |
      renew-start number
      }
    }
```

set (ldap)

```
set pki ldap
    {
    crl-url url_str |
    server-name { name_str | ip_addr }
    }
```

set (pre-prime)

```
set pki pre-prime number
```

set (src-interface)

```
set pki src-interface interface
```

set (x509)

```
set pki x509
    {
    cert-fqdn string |
    default
      {
      cert-path { full | partial }
      crl-refresh { daily | default | monthly | weekly } |
      no-preload-ca |
      send-to string
      } |
    dn
      {
      country-name name_str |
      domain-component string |
      email string |
      ip ip_addr |
      local-name name_str |
      name name_str |
      org-name name_str |
      org-unit-name name_str |
      phone string |
      state-name name_str
      } |
    friendly-name string |
    raw-cn enable |
    renew id_num
    }
```

Keywords and Variables

authentication

```
set pki authority { ... } scep authentication { failed | passed } [ id_num ]
```

authentication	Sets the result of the CA certificate authentication, failed or passed. The *id_num* value identifies a pending certificate created during a SCEP operation.

Example: The following command sets the result of a CA certificate authentication to passed:

set pki authority default scep authentication passed

authority

get pki authority { *id_num* | default } { ... }
set pki authority { *id_num* | default } { ... }
unset pki authority { *id_num* | default } { ... }

authority Defines how the security device uses the CA's authorization services. The *id_num* parameter is the identification number of the CA certificate.

The **default** switch directs the device to use the authority configuration (used when the CA certificate does not reside locally).

Example: The following command instructs the security device to check for certificate revocation on a daily basis:

set pki authority default cert-status crl refresh daily

cert-path

get pki authority { *id_num* | default } cert-path
set pki authority { *id_num* | default } cert-path { full | partial }
unset pki authority *id_num* cert-path

cert-path Defines the X509 certificate path validation level.

When the device verifies a certificate, it builds a certificate chain from certificates received from the peer and the certificate stored locally. Certificates loaded locally are considered trusted.

- **full** Directs the security device to validate the certificate chain to the root. (The last certificate in the certificate chain must be a self-signed CA certificate.)
- **partial** Specifies partial path validation. (The last certificate in the certificate chain may be any locally stored certificate.)

In either case, the last certificate in the chain must come from local storage. You can set this certificate path validation level for a CA.

Example: The following command defines the certificate path validation level as full:

set pki authority default cert-path full

cert-status

get pki authority { *id_num* | default } cert-status
set pki authority { *id_num* | default } cert-status { … }
unset pki authority { *id_num* | default } cert-status { … }

cert-status

Defines how the security device verifies the revocation status of a certificate.

- crl Configures Certificate Revocation List (CRL) parameters.
  - refresh Determines how often (daily, monthly, or weekly) the security device updates the CRL before the CRL expires. The default option uses the validation date decided by the CRL.
  - server-name { *ip_addr:port_num* | *dom_name* } Specifies the server by IP address and port number, or by domain name.
  - url *url_str* Specifies the URL for accessing the CRL.
- ocsp Configures Online Certificate Status Protocol (OCSP) parameters.
  - cert-verify id *number* Identifies the certificate to use when verifying the OCSP response.
  - not-verify-revoke Disables verification of revocation status on the OCSP signing certificate.
  - url *url_str* Specifies the URL for accessing the OCSP responder.
- revocation-check Specifies how the security device checks certificates to see if they are currently revoked.
  - crl Specifies that the device uses CRL to check certificate status.
  - none Specifies that the device does not perform a check of certificate status.
  - ocsp Specifies that the device uses OCSP to check certificate status.
  - best-effort Specifies that the device can use a certificate for which there is no revocation information. This option is useful when CRL retrieval is not practical. For example, in some environments the CRL server is only accessible through a tunnel; however, the CRL information is necessary to build the tunnel originally. When you use the best-effort setting, it is advisable to check the event log periodically. The device should accept a certificate without revocation information only when no revocation information is available. Repeatedly failing to get revocation information for a certificate usually indicates improper configuration.

Example: The following command directs the security device to use the CRL to check certificate status:

set pki authority default cert-status revocation-check crl

cert-verify id

set pki authority *id_num1* cert-status ocsp cert-verify id *id_num2*
unset pki authority *id_num* cert-status ocsp cert-verify

cert-verify id — Identifies a locally stored certificate the security device uses to verify the signature on an OCSP responder.

- *id_num1* Identifies the CA certificate that issued the certificate being verified.
- *id_num2* Identifies the locally stored certificate the device uses to verify the signature on the OCSP response.

convert-cert

exec pki convert-cert

convert-cert — Converts vsys certificate (for versions prior to ScreenOS 3.0.0) to use the internal vsys identifier in ScreenOS 3.0.0 and above.

dsa new-key

exec pki dsa new-key *number*

dsa new-key — Generates a new DSA public/private key pair with a specified bit length (*number*). Key length is 512, 786, 1024, or 2048.

ldap

get pki ldap
set pki ldap { ... }
unset pki ldap { ... }

ldap — Specifies settings for the LDAP server, when the CA certificate associated with the server is not in the device.

- crl-url *url_str* Sets the default LDAP URL for retrieving the certificate revocation list (CRL).
- server-name { *name_str* | *ip_addr:port_num* } Defines the fully qualified domain name or IP address and port number of the server.

Example: The following command assigns 162.128.20.12 as the server's IP address:

```
set pki ldap server-name 162.128.20.12
```

pre-prime

get pki pre-prime
set pki pre-prime *number*
unset pki pre-prime

pre-prime

The **get** command displays:

- The number of precalculated primes for every key type and key length combination. The key type can be DSA or RSA and the key length can be 1024 or 2048 bits depending on the device platform.

  Note: Juniper Networks security appliances generate 1024-bit primes. Juniper Networks systems generate 1024-bit and 2048-bit primes. For more information, refer to the datasheet for your security device.
- The number of currently available pairs of prime numbers for every key type and key length combination.
- Ongoing prime calculation for a key-type and key-length combination and the number of attempts already made.

The **set** command instructs the security device to generate a specific number of precalculated primes to store in memory.

The **unset** command reverts the security device to the default number of precalculated primes. The default number of precalculated primes is platform-specific. For more information, refer to the datasheet for your security device.

rsa new-key

exec pki rsa new-key *number* [&]

rsa new-key

Generates a new RSA public/private key pair with a specified bit length (*number*). Key length is 512, 786, 1024, or 2048.

The **&** switch directs the device to perform key generation in the background, without waiting for the result. Without this switch, the device waits up to 100 seconds.

scep

exec pki x509 scep { cert *id_num* | key { *id_num* | last-key } | renew }
get pki authority { *id_num* | default } scep
set pki authority { *id_num* | default } scep { ... }
unset pki authority { *id_num* | default } scep { ... }

scep

Defines Simple Certificate Enrollment Protocol (SCEP) parameters.

- authentication { passed | failed } [*id_num*] sets the result of the CA authentication, failed or passed. The *id_num* value identifies a defined key pair.
- ca-cgi *url_str* Specifies the path to the CA's SCEP server.
- ca-id *string* Specifies the identity of the CA's SCEP server.
- cert-id *id_num* Directs the security device to retrieve the final certificate for a pending certification.
- challenge *pswd_str* Specifies the Challenge password.
- current Directs the security device to use the SCEP associated with a CA as default.
- key *id_num* Directs the device to acquire a certificate for the specified key pair. The *id_num* parameter specifies the ID of a specific key pair. The last_key parameter specifies the most recently created key pair.
- mode { auto | manual } Specifies the authentication mode to authenticate the certificate.
- polling-int *number* Determines the retrieval polling interval (in minutes). The default value is 0 (none).
- ra-cgi *url_str* Specifies the CGI path to the RA's SCEP server.
- renew *id_num* Directs the device to renew the specified certificate (*id_num*).
- renew-start Set the number of days before the certificate expiration date when you want the security device to request the renewal of the certificate.

Example: The following command sets the SCEP Challenge password to “swordfish”:

```
set pki authority default scep challenge swordfish
```

Example: The following command uses the SCEP setting for CA 123 as the default:

```
set pki authority 123 scep current
```

send-to

```
get pki x509 send-to
set pki x509 default send-to string
unset pki x509 default send-to
```

send-to Specifies or displays the email destination (*string*) to send the x509 certificate request file.

src-interface

```
get pki src-interface
set pki src-interface
unset pki src-interface
```

src-interface Displays, configures or removes the source interface the security device uses to send PKI traffic.

x509

```
exec pki x509 { ... }
get pki x509 { ... }
set pki x509 { ... }
unset pki x509 { ... }
```

x509 Specifies settings for x509 certificates, displays certificate information, and performs various operations related to x509 PKI object.

- cert *id_num* Displays information about the specified certificate.
- cert-fqdn *string* Configures the fully qualified domain name (FQDN). PKI uses this value in the certificate subject alt name extension.
- default Specifies settings for the CA whose certificate is not locally configured.
  - crl-refresh Sets or displays the refreshment frequency (daily, monthly, or weekly) of the X.509 CRL. The default option uses the expiration date in each CRL.
  - send-to *string* Assigns the email address to which the security device sends the PKCS10 certificate request file.
- dn Specifies or displays the name that uniquely identifies a requesting certificate.
  - country-name *name_str* Sets the country name.
  - email *string* Sets the email address.
  - ip *ip_addr* Sets the IPv4 or IPv6 address.
  - local-name *string* Sets the locality.
  - name *string* Sets the name in a common name field.
  - org-name *string* Sets the organization name.
  - org-unit-name *string* Sets the organization unit name.
  - phone *string* Sets a contact phone number as the X.509 certificate subject name of the security device.
  - state-name *string* Sets the state name as the X.509 certificate subject name.

- friendly-name *name_str id_num* A friendly name (*name_str*) for the certificate (*id_num*).
- install-factory-certs *name_str* Loads a specified factory predefined certificate.
- list Displays the X.509 object list.
  - ca-cert Displays all CA certificates.
  - cert Displays all X.509 certificates.
  - key-pair Displays all key pairs for which there is no certificate.
  - crl Displays all Certificate Revocation Lists (CRLs).
  - local-cert Displays all local certificates.
  - pending-cert Displays all pending certificates.
- pkcs10 Displays a PKCS10 file (an X.509 certificate request) for a key pair.
- raw-cn enable Enables the raw common name (CN) or displays its current status.
- scep
  - cert *id_num* Initiates Simple Certificate Enrollment Protocol (SCEP) operation to retrieve certificates from a CA server. The *id_num* parameter is the identification number of the pending certificate.
  - key { *id_num* | last-key } Initiates SCEP operation to obtain a certificate for a key pair. The variable *id_num* identifies the key pair and last-key specifies to obtain a certificate for the most recently created key pair.
  - renew *id_num* Initiates SCEP operation to renew an existing certificate. The variable *id_num* identifies the existing certificate to renew.
- tftp *ip_addr* Uploads the specified certificate (cert-name *name_str*) or CRL file (crl-name *name_str*) for the specified TFTP server at an IPv4 or IPv6 address *ip_addr*.

Examples: The following command specifies the destination email address where the security device sends the PKCS10 certificate request:

```
set pki x509 default send-to caServer@somewhere.com
```

The following command refreshes the certificate revocation list on a daily basis:

```
set pki x509 default crl-refresh daily
```

The following command defines a distinguished name for Ed Jones, who works in marketing at Juniper Networks in Sunnyvale, California:

```
set pki x509 dn country-name US
set pki x509 dn state-name CA
set pki x509 dn local-name Sunnyvale
set pki x509 dn org-name “Juniper Networks”
set pki x509 dn org-unit-name marketing
set pki x509 dn name “Ed Jones”
```

Defaults

The RSA key length is set to 1024 bits.

Requesting a CA Certificate

You use the set pki, get pki, and exec pki commands to request an x509 CA certificate from a certificate authority. The following commands provide a typical example:

1. Specify a CA CGI path.

   set pki auth default scep ca-cgi "http://pilotonsiteipsec.verisign.com/cgi-bin/pkiclient.exe"

NOTE: The Common Gateway Interface (CGI) is a standard way for a web server to pass a user request to an application program, and to receive data back. CGI is part of HyperText Transfer Protocol (HTTP).

2. Specify a registration authority RA CGI path

   set pki auth default scep ra-cgi "http://pilotonsiteipsec.verisign.com/cgi-bin/pkiclient.exe"

NOTE: You must specify an RA CGI path even if the RA does not exist. If the RA does not exist, use the value specified for the CA CGI.

3. Generate an RSA key pair, specifying a key length of 1024 bits.

   exec pki rsa new 1024

4. Initiate the SCEP operation to request a local certificate.

   exec pki x509 scep key last-key

5. If this is the first attempt to apply for a certificate from this CA, a prompt appears presenting a fingerprint value for the CA certificate. (Otherwise, go to step 6.)

   After verification of the fingerprint, allow the operation to continue by executing the following command:

   set pki auth default scep auth passed

   You must specify an RA CGI path even if the RA does not exist. If the RA does not exist, use the value specified for the CA CGI.

6. If the device does not approve the certificate automatically, contact your CA administrator to approve the local certificate request.

7. (Optional) Display a list of pending certificates. This allows you to see and record the ID number identifying the pending certificate.

   get pki x509 list pending-cert

8. (Optional) Obtain the local certificate from the CA (using the ID number obtained in step 7) to identify the certificate. In this example, the certificate number is 1001.

   ```
   exec pki x509 scep cert 1001
   ```

policy

Use the **policy** commands to define policies to control network and VPN traffic.

A *policy* is a set of rules that determines how traffic passes between security zones (interzone policy), between interfaces bound to the same zone (intrazone policy), and between addresses in the Global zone (global policy). When a security device attempts to pass a packet from one zone to another, between two interfaces bound to the same zone, or between two addresses in the Global zone, the security device checks its policy lists for a policy to permit such traffic. For example, to allow traffic to pass from one security zone to another, you must configure a policy that permits zone A to send traffic to zone B. To allow traffic originating in zone B to flow to zone A, you must configure another policy permitting traffic from zone B to zone A.

Executing the **set policy id** *pol_num* command without specifying further options places the CLI within the context of an existing policy. For example, the following commands define a policy with ID number 1 and then enter the “policy:1” context to add a second service:

```
device-> set policy id 1 from trust to untrust host1 host2 HTTP permit
device-> set policy id 1
device(policy:1)-> set service FTP
```

After you enter a policy context, all subsequent command executions modify the specified policy (policy:1 in this example). To save your changes, you must first exit the policy context, and then enter the **save** command:

```
device(policy:1)-> exit
device-> save
```

You can also use the **set policy id** *pol_num* command with additional options to modify an existing policy. For example, the following commands add a Deep Inspection extension to policy 1:

```
device-> set policy id 1 from trust to untrust host1 host2 HTTP permit
device-> set policy id 1 attack HIGH:HTTP:SIGS action close
```

NOTE: The above example adds a deep inspection (DI) extension that was not present in the original policy. After you enter a policy context, you cannot add a DI extension if one does not already exist in the original policy.

Syntax

exec

```
exec policy verify [ from zone [ to zone ] | global | to zone ]
```

get

```
get policy
    [
    all |
    from zone1 to zone2 |
    [ global ] id pol_num
    ]
```

get (Within a Policy Context)

```
get configuration
```

set

```
set policy
    [ global ]
    [ id pol_num1 ]
    [ top | before pol_num2 ]
    [ name name_str ]
    [ from zone1 to zone2 ]
    src_addr dst_addr svc_name
      [
      nat
        [ src [ dip-id id_num ] ]
          [ dst ip addr1 [ addr2 | port port_num ] ]
      ]
        {
        permit |
        tunnel { l2tp tunn_str | vpn-group id_num } |
        tunnel vpn tunn_str [ l2tp tunn_str | pair-policy pol_num ]
        }
        [ no-hw-sess ]
          [ auth [ server name_str ] | webauth ]
            [
            group-expression string |
            user name_str | user-group name_str
            ]
          ] |
        deny
            [ schedule name_str ]
            [ log [ alert ] ]
              [ count [ alarm id_num1 id_num2 ] ]
                [ no-session-backup ]
            [ url-filter ]
                  [ traffic { gbw number }
                    { priority number }
                      { mbw [ number ]
                        dscp { disable | enable }
                      }
                  ]
            [ attack string action string | av name_str ]
    }
set policy move pol_num1 { before pol_num2 | after pol_num3 }
set policy default-permit-all
```

set policy id number

```
set policy [ global ] id pol_num disable
set policy [ global ] id pol_num application svc_name
set policy [ global ] id pol_num attack string action string
set policy [ global ] id pol_num av name_str
```

set (Within a Policy Context)

```
set
    {
    attack string |
    av name_str |
    count [ alarm number1 number2 ] |
    di-alert-disable |
    di-severity { info | low | medium | high | critical } |
    dst-address | src-address
      { name_str | negate } |
    log [ alert ] |
    name name_str |
    no-hw-sess |
    service svc_name |
    src-address { name_str | negate }
    }
```

unset (Within a Policy Context)

```
unset attack string
unset av name_str
unset count
unset { dst-address | src-address } { name_str | negate }
unset { ims-alert | ims-log }
unset log
unset name
unset service svc_name
unset severity
```

Keywords and Variables

all

```
get policy all
```

all	Displays information about all security policies.

application

set policy [global] id *pol_num* application *svc_name*

application	Defines the type of Layer 7 application associated with a Layer 3 service and Layer 4 port number. This is particularly important for defining the Layer 7 application for custom services so that the security device can properly inspect such traffic for attack signatures and anomalies. The **ignore** option, which appears near the end of the list of application choices, instructs the security device to ignore the application type typically associated with a predefined service and port number. Using the **ignore** option instructs the security device not to scan the packet payload and can prevent the security device from attempting to parse one type of traffic when it is actually another type—such as the case with LDAP and H.323 traffic, both of which use TCP port 389. The **none** option, which also appears near the end of the list of application choices, instructs the security device to use the default setting. Choosing **none** is the equivalent to entering the CLI command: **unset policy id** *id_num* **application**.

Example: The following command identifies the Layer 7 application for policy ID 1 as FTP:

set policy id 1 application FTP

attack

set policy { … } attack *string* action *string*
set attack *string*
unset policy { *pol_num* **|** id *pol_num* } attack
unset attack *string*

attack *string*	Inspects traffic to which the policy applies for attack objects in the specified attack object group. Attack objects can be stateful signatures or protocol anomalies. If the security device detects an attack object, it then performs the specified **action**, which can be one of the following: ■ **Close** The security device logs the event, severs the connection, and sends TCP RST packets to both the client and server. ■ **Close Client** The security device logs the event, severs the connection, and sends a TCP RST packet to the client. ■ **Close Server** The security device logs the event, severs the connection, and sends a TCP RST to the server. ■ **Drop** The security device logs the event and severs the connection without sending either the client or the server TCP RST packets. ■ **Drop Packet** The security device logs the event and drops the packet containing the attack object, but it does not sever the connection. ■ **Ignore** The security device logs the event and stops checking—or ignores—the remainder of the connection. ■ **None** The security device logs the event but takes no action.

Example: The following commands define a policy to check for attack objects in the CRITICAL:HTTP:ANOM, CRITICAL:HTTP:SIGS, HIGH:HTTP:ANOM, and HIGH:HTTP:SIGS attack object groups in HTTP traffic from any host in the Untrust zone to webserver1 in the DMZ zone. If the security device detects any attack objects, it then severs the connection and sends webserver1 a TCP RST so it can clear its resources:

```
device-> set policy id 1 from untrust to dmz any webserver1 http permit attack
    CRITICAL:HTTP:ANOM action close server
device-> set policy id 1
device(policy:1)-> set attack CRITICAL:HTTP:SIGS
device(policy:1)-> set attack HIGH:HTTP:ANOM
device(policy:1)-> set attack HIGH:HTTP:SIGS
```

av

```
set policy { … } av name_str
set av name_str
unset policy { pol_num | id pol_num } av name_str
unset av name_str
```

av *name_str* Sends HTTP or SMTP traffic to which the policy applies to the specified antivirus (AV) scanner, which examines the data for viruses. If it finds a virus, the AV scanner quarantines the infected data for further study and returns the SMTP or HTTP file—without the infected data—to the security device, which then forwards the file to the intended recipient

Example: The following command instructs the security device to forward SMTP traffic originating from the remote mail server “r-mail1” in the Untrust zone and destined for the local mail server “mail1” in the DMZ zone to an AV scanner named “av1”:

```
set policy id 1 from untrust to dmz r-mail1 mail1 smtp permit av av1
```

auth

```
set policy { ... } auth [ … ]
```

auth Requires the user to provide a login name and password to authenticate his or her identity before accessing the device and crossing the firewall.

- server *name_str* Identifies the authentication server (*name_str*).
- group-expression *string* Identifies users according to an expression (*string*).
- user *name_str* Identifies a user (*name_str*).
- user-group *name_str* Identifies a user group (*name_str*).

Example: The following command invokes user authentication.

- Permits any kind of traffic from any address in the Trust zone to any address in the Untrust zone
- Uses an authentication server named wc-server

```
set policy from trust to untrust any any any permit auth server wc-server
```

before

set policy before *pol_num1* { ... }

before	Specifies the position of the policy before another policy (*pol_num*) in the access control list (ACL).

Example: The following command creates a new policy with ID number 3 and positions it before the policy with ID number 2:

```
set policy id 3 before 2 from trust to untrust any any any permit
```

configuration

get configuration

configuration	Displays the configuration details for the policy in whose context you issue the **get configuration** command.

count

set policy { ... } [count [alarm { *id_num1 id_num2* }]] { ... }

count	Maintains a count, in bytes, of all the network traffic the policy allows to pass through the security device. The **alarm** *number1 number2* parameter enables the alarm feature so that you can view alarms. You must enter the number of bytes per second (*number1*) and the number of bytes per minute (*number2*) required to trigger an alarm.

Example: The following command permits any kind of traffic from any address in the Trust zone to any address in the Untrust zone and maintains a count of all network traffic to which the policy applies:

```
set policy from trust to untrust any any any permit count
```

default-permit-all

set policy default-permit-all

default-permit-all	Allows access without checking the access control list (ACL) for a matching policy.

disable

set policy [global] id *pol_num* disable

disable	Disables the policy without removing it from the configuration.

di-alert-disable (within a policy context)

set di-alert-disable

di-alert-disable	Disables DI (Deep Inspection) alert generation.

di-severity (within a policy context)

set di-severity

disable

Specifies the severity of events that generate error messages. The possible event levels are info, low, medium, high, and critical.

from ... to

set policy { ... } from *zone1* to *zone2 src_addr dst_addr svc_name* { ... } [...]

from *zone1* to *zone2*
src_addr dst_addr
svc_name

Specifies two zones between which a policy controls traffic.

- *zone1* is the name of the source security zone.
- *zone2* is the name of the destination security zone.
- *src_addr* is the name of the source address. Specifying any allows all source IP addresses.
- *dst_addr* is the name of the destination address. Specifying any allows all destination IP addresses.
- *svc_name* is the name of the service. Specifying any identifies all available services.

For more information, see “Zones” on page 321.

Example: The following command permits HTTP traffic from any address in the Trust zone to any address in the Untrust zone:

set policy from trust to untrust any any HTTP permit

global

set policy global before { ... }

set policy global id *pol_num* disable

set policy global move *pol_num1* { before *pol_num2* **|** after *pol_num3* }

set policy global name *name_str* { ... }

set policy global top

global

Creates or displays policies that use the Global zone. The Global zone address book keeps all the VIPs of all interfaces, regardless of the zone to which the interface belongs. You can use these VIP addresses as destination addresses in policies between any two security zones.

id

get policy [global] id *pol_num*

set policy [global] id *pol_num1* { ... }

unset policy id *pol_num* [disable]

id *pol_num*

Specifies an policy ID number. (The disable switch disables the policy.)

Example: The following command assigns the policy an ID value of 10 and permits FTP-GET traffic from any address in the Trust zone to any address in the Untrust zone:

set policy id 10 from trust to untrust any any ftp-get permit

idp-alert-disable

```
set idp-alert-disable
unset idp-alert-disable
```

idp-alert-disable	Disables the syslog alert feature for any detected attack.

l2tp

set policy [global] { ... } tunnel l2tp *tunn_str* { ... }
set policy [global] { ... } tunnel vpn *tunn_str* l2tp *tunn_str*

l2tp	Specifies a Layer 2 Tunneling Protocol (L2TP) tunnel.

Example: The following command defines an inbound policy for an L2TP tunnel.

- VPN tunnel named “home2office”
- L2TP tunnel named “home-office”
- Dialup VPN group named “home_office”

set policy from untrust to trust dialup_vpn our_side any tunnel vpn home2office l2tp home_office

log

set policy [global] { ... } log [alert] { ... }

log [alert]	Maintains a log of all connections to which the policy applies. The alert switch enables the syslog alert feature.

Example: The following command creates a policy and directs the security device to log the traffic to which the policy applies.

- Permits HTTP traffic from any address in the Trust zone to any address in the Untrust zone
- Directs the security device to log the traffic to which the policy applies
- Enables the syslog alert feature

set policy from trust to untrust any any HTTP permit log alert

move

set policy [global] move *pol_num1* { before *pol_num2* **|** after *pol_num3* }

move	Repositions a policy (*pol_num1*) before another policy (*pol_num2*) or after a policy (*pol_num3*) in the access control list (ACL). When one policy comes before another policy in the ACL, it has higher precedence.

Example: The following command positions a global policy with ID number 4 before the policy with ID number 2:

set policy global move 4 before 2

name

set policy [global] [...] name *name_str* {... }

name *name_str*	Identifies the policy by name. (Assigning a name to an policy is optional.)

Example: The following command creates a new policy named “outbound”:

set policy name outbound from trust to untrust any any any permit

nat

set policy [global] { ... } nat src [dip-id *id_num*] { ... }
set policy [global] { ... } nat dst ip *addr1* [*addr2* **|** port *port_num*] { ... }

nat

Enables or disables source and destination Network Address Translation (NAT-src and NAT-dst). This feature translates the original source or destination IP address in an IP packet header to another address.

- src Performs NAT-src on traffic to which the policy applies. The security device can perform NAT-src with addresses from a Dynamic IP (DIP) pool, or using the egress interface IP address (in which case, you do not specify a DIP pool).
  - dip-id *id_num* Specifies the ID number of a DIP pool. This number can be between 4 and 255.
- dst Performs NAT-dst on traffic to which the policy applies. ScreenOS supports the following three options for NAT-dst:
  - ip *addr1* Translates the original destination address to the address specified in the policy. The security device does not translate the original port number.
  - ip *addr1 addr2* Translates the original destination IP address from one range of addresses to an address in another range of addresses. The security device maintains a consistent mapping of an original destination address to a translated address within the specified range using a technique called address shifting.
  - ip *addr1* port *port_num* Translates the original destination address and port number to the address and port number specified in the policy.

Example1: The following command creates a policy that applies NAT-src on all traffic from any address in the Trust zone to any address in the Untrust zone and specifies DIP pool 8:

set policy from trust to untrust any any any nat src dip-id 8 permit

Example2: The following commands create an address (1.1.1.5/32) named v-addr1 in the DMZ zone and a policy that applies NAT-dst on HTTP traffic from any address in the Untrust zone to the virtual destination address v-addr1 in the DMZ zone. The security device translates the destination address from 1.1.1.5 to 10.2.2.5:

set address dmz v-addr1 1.1.1.5/32
set policy from untrust to dmz any v-addr1 http nat dst ip 10.2.2.5 permit

Example3: The following command combines NAT src (source) and dst (destination):

set policy from trust to untrust any any any nat src dip-id 8 dst ip 10.2.2.5 permit

negate

set { dst-address | src-address } negate

negate	Applies the policy in whose context you issue this command to all addresses except those specified as either the destination (**dst-address**) or source (**src-address**). The negate option takes effect at the policy component level, applying to all items in the negated component

Example: The following commands permit HTTP traffic to the Untrust zone from all addresses in the Trust zone except from addr1:

```
device-> set policy id 1 from trust to untrust any any http permit
device-> set policy id 2 from trust to untrust addr1 any http permit
device-> set policy id 2
device(policy:2)-> set src-address negate
```

no-hw-sess

set policy [global] { ... } no-hw-sess { ... }
set no-hw-sess
unset no-hw-sess

no-hw-sess	Disables the security device from creating a hardware session for a specific traffic. This is useful when some traffic could not be handled efficiently by ASIC and for debug operation.

NOTE: For TCP traffic, you must create a dummy hardware session to pass the traffic to the CPU.

no-session-backup

set policy [global] { ... } no-session-backup { ... }

no-session-backup	Disables backing up the sessions to which the policy applies when the security device is in a high availability (HA) configuration. By default, a security device operating in HA backs up sessions.

pair-policy

set policy [global] { ... } pair-policy *pol_num* [...]

pair-policy *pol_num*	Links the policy that you are configuring with another policy that references the same VPN tunnel so that both policies share one proxy ID and one security association (SA). This is useful when you want to allow bidirectional traffic over a policy-based VPN and there is source destination address translation using a DIP pool or destination address translation using a MIP or VIP. Without policy pairing, the security device derives a different proxy ID from both the outbound and inbound policies. This causes a problem for the remote peer if it has only a single proxy ID for the VPN tunnel. By pairing both policies together, they share a single proxy ID (derived from the policy that you configured last), which solves the proxy ID problem for the remote peer, and they share a single SA, which conserves SA resources.

Example: The following commands create two policies sharing the same VPN tunnel and then bind them into a policy pair. (You have previously created on the tunnel interface subnet a DIP pool with ID 4 and addresses 1.1.1.10 – 1.1.1.20, and a MIP from 1.1.1.5 to host 10.1.1.5.):

```
set policy id 1 from trust to untrust addr1 addr2 any nat src dip-id 4 tunnel vpn vpn1
set policy id 2 from untrust to trust addr2 mip(1.1.1.5) MAIL tunnel vpn vpn1
    pair-policy 1
```

The proxy ID for both of these policies is as follows:

```
local 1.1.1.5/255.255.255.255, remote 10.2.2.0/255.255.255.0, proto 6, port 25
```

Because the local address in the above proxy ID does not include the addresses in the DIP pool or any service other than SMTP (or “MAIL”), you must also set a proxy ID with an address range that encompasses both the MIP (1.1.1.5) and DIP pool (1.1.1.10–1.1.1.20) and change the service to “ANY”:

```
set vpn vpn1 proxy-id local-ip 1.1.1.0/24 remote-ip 10.2.2.0/24 ANY
```

deny | nat | permit | reject | tunnel

set policy [global] { ... } | deny | nat | permit | reject | tunnel | [...]

deny | nat | permit | reject | tunnel	■ deny blocks the service at the firewall. The security device simply drops the packet. ■ nat enables Network Address Translation. ■ permit allows the specified service to pass from the source address across the firewall to the destination address. ■ reject blocks the service at the firewall. The security device drops the packet and sends a TCP reset (RST) segment to the source host for TCP traffic and an ICMP “destination unreachable, port unreachable” message (type 3, code 3) for UDP traffic. For types of traffic other than TCP and UDP, the security device drops the packet without notifying the source host, which is also what occurs when the action is “deny”. ■ tunnel allows you to use VPN or IPSEC tunnel.

Example: The following command:

- Defines a policy from the Trust zone to the Untrust zone
- Uses any source or destination IP address
- Permits any kind of service

```
set policy from trust to untrust any any any permit
```

schedule

set policy [global] { ... } schedule *name_str* [...]

schedule	Applies the policy only at times defined in the specified schedule.

Example: With following commands, you first create a schedule named “Mkt_Sched” and then reference it in a policy permitting any kind of traffic from any address in the Trust zone to any address in the Untrust zone:

```
set schedule Mkt_Sched recurrent monday start 09:00 stop 12:00
set policy from trust to untrust any any any permit schedule Mkt_Sched
```

top

```
set policy [ global ] [ ... ] top
```

top	Places the policy at the top of the access control list (ACL). The policy at the top of the ACL has the highest precedence.

Example: The following command:

- Permits any kind of service from any address in the Trust zone to any address in the Untrust zone
- Assigns to the policy an ID value of 30
- Places the policy at the top of the ACL

```
set policy id 30 top from trust to untrust any any any permit
```

traffic gbw

```
set policy [ global ] [ ... ] traffic
    gbw number priority number mbw [ number ] dscp { disable | enable }
```

traffic gbw — Defines the guaranteed bandwidth (GBW) in kilobits per second. The security device passes traffic below this threshold with the highest priority, without performing traffic shaping.

- priority *number* Specifies one of the eight traffic priority levels. When traffic falls between the guaranteed and maximum bandwidth settings, the security device passes traffic with higher priority first. Lower priority traffic is passed only if there is no higher priority traffic.
- mbw *number* Defines the maximum bandwidth (MBW) in kilobits per second. Traffic beyond this limit is throttled and dropped.
- dscp { enable | disable } Enables or disables a mapping of the eight ScreenOS priority levels to the Differentiated Services—DiffServ—Codepoint (DSCP) marking system. In the ScreenOS system, 0 is the highest priority, and 7 is the lowest.

Example: The following command performs these tasks:

- Permits HTTP traffic from any address in the Trust zone to any address in the Untrust zone
- Guarantees bandwidth of 3,000 kilobits per second
- Assigns a priority value of 2
- Sets the maximum bandwidth to 10,000 kilobits per second
- Enables mapping of the eight ScreenOS priority levels to the DiffServ Codepoint (DSCP) marking system

```
set policy from trust to untrust any any HTTP permit traffic gbw 3000 priority 2 mbw
    10000 dscp enable
```

tunnel

```
set policy [ global ] { ... } tunnel
      { l2tp tunn_str | vpn-group id_num }
set policy [ global ] { ... } tunnel vpn tunn_str
      [ l2tp tunn_str | pair-policy pol_num ]
```

tunnel

Encrypts outgoing IP packets, and decrypts incoming IP packets.

- **vpn** [**l2tp** *tunn_str*] Identifies a VPN tunnel. For an IPSec VPN tunnel, specify **vpn** and the name of the VPN tunnel. For L2TP, specify **vpn** (with the name of the VPN tunnel) and **l2tp** (with the name of the L2TP tunnel).
- **vpn** [**pair-policy** *id_num*] Links this policy with an existing policy also referencing the same VPN. The VPN uses the proxy-id derived from the policy whose configuration includes the **pair-policy** keyword.
- **vpn-group** *id_num* Identifies a VPN group (*id_num*). A VPN group consist of multiple VPNs, which you can specify in a single policy.
- **vpn-tunnel** Identifies an active tunnel.

Example: The following command defines a policy that uses a defined VPN tunnel.

- Encrypts traffic exchanged with the corporate headquarters (denoted by address-book entry Headquarters)
- Uses a VPN named To_HQ:

```
set policy from trust to untrust any Headquarters any tunnel vpn To_HQ
```

verify

```
exec policy verify [ from zone [ to zone ] | global | to zone ]
```

verify

Verifies that the order of policies in a policy list is valid so that a policy higher in the list does not eclipse, or “shadow”, another policy lower in the list. If the verification check discovers policy shadowing, the command output explains which policies are shadowing which. You can define the scope of the verification as follows:

- Not setting any further options instructs the security device to verify the ordering of policies in all policy sets.
- **from** *zone* Checks the ordering of policies from the specified zone to any zone.
- **from** *zone* **to** *zone* Checks the ordering of policies between the specified zones.
- **global** Checks the ordering of policies in the global policy set.
- **to** *zone* Checks the ordering of policies from any zone to the specified zone.

Example: The following command verifies the ordering of policies from the Trust zone to the Untrust zone:

```
exec policy verify from trust to untrust
```

port-mode

Use the **port-mode** commands to set the port, interface, and zone bindings for the security device. (Use the **get system** command to see the current port mode setting.)

NOTE: Setting the port mode removes any existing configurations on the device and requires a system reset.

Syntax

```
exec port-mode { trust-untrust | home-work | dual-untrust | combined }
```

Keywords and Variables

trust-untrust

Defines the following port, interface, and zone bindings:

- Binds the Untrusted Ethernet port to the untrust interface, which is bound to the Untrust zone.
- Binds the Trusted1 through Trusted4 Ethernet ports to the trust interface, which is bound to the Trust zone.
- Binds the Modem port to the serial interface, which you can bind as a backup interface to the Untrust zone.

This is the default port mode.

home-work

Defines the following port, interface, and zone bindings:

- Binds the Untrusted Ethernet port to the ethernet3 interface, which is bound to the Unturst zone.
- Binds the Trusted4 and Trusted3 Ethernet ports to the ethernet2 interface, which is bound to the Home zone.
- Binds the Trusted2 and Trusted1 Ethernet ports to the ethernet1 interface, which is bound to the Work zone.
- Binds the Modem port to the serial interface, which you can bind as a backup interface to the Untrust zone.

dual-untrust

Defines the following port, interface, and zone bindings:

- Binds the Untrusted port to the ethernet3 interface, which is bound to the Untrust zone.
- Binds the Trusted4 Ethernet port to the ethernet2 interface, which is bound as a backup interface to the Untrust zone.
- Binds the Trusted1 through Trusted3 Ethernet ports to the ethernet1 interface, which is bound to the Trust zone.

combined

Defines the following port, interface, and zone bindings:

- Binds the Untrusted Ethernet port to the ethernet4 interface, which is bound to the Untrust zone.
- Binds the Trusted4 Ethernet port to the ethernet3 interface, which is bound as a backup interface to the Untrust zone.
- Binds the Trusted3 and Trusted2 Ethernet ports to the ethernet2 interface, which is bound to the Home zone.
- Binds the Trusted1 Ethernet port to the ethernet1 interface, which is bound to the Work zone.

The combined port mode is supported only on the NetScreen-5XT Elite (unrestricted users) platform.

NOTE: Setting the port mode removes any existing configurations on the device and requires a system reset.

pppoe

Use the pppoe commands to configure PPPoE or to display current PPPoE configuration parameters.

Point-to-Point Protocol over Ethernet (PPPoE) is a protocol that allows the members of an Ethernet LAN to make individual PPP connections with their ISP by encapsulating the IP packet within the PPP payload, which is encapsulated inside the PPPoE payload. Some security devices support PPPoE, which allows them to operate compatibly on DSL, Ethernet Direct, and cable networks run by ISPs that use PPPoE to give their clients Internet access.

Syntax

clear

```
clear [ cluster ] pppoe [ name name_str ]
```

exec

```
exec pppoe [ name name_str ] { connect | disconnect }
```

get

```
get pppoe
    [
    all |
    name name_str | id id_num
      [ configuration | statistics ]
    ]
```

set

```
set pppoe [ name name_str ]
    {
    ac name_str |
    authentication { CHAP | PAP | any } |
    auto-connect number |
    clear-on-disconnect |
    default-route-metric number |
    enable |
    idle-interval number |
    interface [ name_str ] |
    name-server admin-preference |
    netmask mask |
    ppp
      {
      lcp-echo-retries number |
      lcp-echo-timeout number
      } |
    service name_str |
    static-ip |
    update-dhcpserver |
    username name_str password pswd_str
    }
```

Keywords and Variables

ac

set pppoe ac *name_str*
unset pppoe ac

ac	Allows the interface to connect only to the specified AC (access concentrator).

all

get pppoe all

all	Displays information for all PPPoE instances.

authentication

set pppoe authentication { CHAP **|** PAP **|** any }
unset pppoe authentication { CHAP **|** PAP }

authentication	Sets the authentication methods to CHAP, PAP, or any. (The any option gives preference to CHAP.) The default of authentication is any (both CHAP and PAP). To set authentication to CHAP only, first execute **unset pppoe authentication PAP**.

auto-connect

set pppoe auto-connect *number*
unset pppoe auto-connect

auto-connect	Specifies the number of seconds that elapse before automatic re-initiation of a previously closed connection occurs. Valid range is 0-10000. (0 to disable.)

clear-on-disconnect

set pppoe [name *name_str*] clear-on-disconnect
unset pppoe clear-on-disconnect

clear-on-disconnect	Directs the security device to clear the IP address and the gateway for the interface once PPPoE disconnects. By default, this is disabled; that is, the IP address and gateway for the interface remain when PPPoE disconnects. If you do not specify **name**, ScreenOS sets the parameter for the default instance untrust.

cluster

clear cluster pppoe

cluster	Propagates the **clear** operation to all other devices in an NSRP cluster.

configuration

get pppoe [**name** *name_str*] **configuration**

configuration — Displays the configuration options.

If you do not specify name, ScreenOS displays the parameters for the default instance untrust.

connect | disconnect

exec pppoe [name *name_str*] { connect | disconnect }

connect — Starts a PPPoE connection for an instance. (Each instance can be bound to an interface.)

disconnect — Takes down a PPPoE connection.

default-route-metric

set pppoe default-route-metric *number*
unset pppoe default-route-metric

default-route-metric — Sets the metric for the default route for the current instance.

enable

set pppoe [name *name_str*] enable
unset pppoe [name *name_str*] enable

enable — Enables or disables a PPPoE instance, without removing the object that defines the instance. This allows you to temporarily disable the instance, and enable it later without redefining it.

idle-interval

set pppoe idle-interval *number*
unset pppoe idle-interval

idle-interval — Sets the idle timeout, which is time elapsed (in minutes) before the security device terminates a PPPoE connection due to inactivity. Specifying 0 turns off the idle timeout and the device never terminates the connection.

id

get pppoe id *id_num*

id — Specifies a PPPoE instance by ID number.

interface

set pppoe interface [*name_str*]
unset pppoe interface

interface — Specifies the interface for PPPoE encapsulation.

name

exec pppoe [name *name_str*] { connect **|** disconnect }
get pppoe [name *name_str* **|** all]
set pppoe [name *name_str*] ...
unset pppoe [name *name_str*]

name	Specifies or defines the name for a specific PPPoE instance. You can assign a username and password, interface, and other PPP/PPPoE parameters to the instance. If you do not specify name, ScreenOS automatically configures the parameters for the default instance untrust.

Example: The following commands define a name for a PPPoE instance.

- Username *user1* and password *123456*
- PPPoE instance pppoe-user-1 bound to the *ethernet2* interface

```
set pppoe name pppoe-user-1 username user1 password 123456
set pppoe name pppoe-user-1 interface ethernet2
```

name-server

set pppoe name-server admin-preference *number*
unset pppoe name-server admin-preference

name-server	Specifies the preference level for DNS addresses learned from the PPPoE server. The device can learn DNS server addresses statically (from the CLI or WebUI), or it can learn them dynamically (from PPPoE, DHCPv6, DHCP or XAuth). The device stores these learned addresses in the DNS server list. It then selects the best two addresses from this list, and designates them as the primary and secondary DNS server addresses. The admin-preference number setting specifies how much preference the device gives to addresses learned through one source or protocol, in comparison with another source or protocol. To do this, it uses an election protocol. First, the device compares the admin-preference values. If the values differ, it selects the address with the highest value. If the values are identical, it uses the highest protocol. (The protocol levels, from highest to lowest, are PPPoE, XAuth, DHCP, and CLI, respectively.) If the protocols are identical, it chooses the address with the greatest numerical value.

netmask

set pppoe netmask *mask*
unset pppoe netmask

netmask	Specifies a PPPoE subnet mask that the device assigns to the interface bound to the PPPoE instance (after establishment of the connection). When it is necessary for two or more interfaces to have overlapping subnets, use the following command: set vrouter *vrouter* ignore-subnet-conflict

ppp

set pppoe ppp { … }
unset pppoe ppp { … }

ppp — Specifies PPP parameters.

- **lcp-echo-retries** the number of unacknowledged Link Control Protocol (LCP) Echo requests before connection is terminated. Valid range is 1-30.
- **lcp-echo-timeout** the time that elapses between transmission of two LCP Echo requests. Valid range is 1-1000 seconds.

service

set pppoe service *name_str*
unset pppoe service

service — Allows only the specified service (*name_str*). This feature uses service tags to enable a PPP over Ethernet (PPPoE) server to offer PPPoE clients a selection of services during call setup. The user can choose an offered service, and the security device provides the service when the PPPoE session becomes active. This allows service providers to offer services and to charge customers according to the service chosen.

static-ip

set pppoe static-ip
unset pppoe static-ip

static-ip — Specifies that your connection uses the IP address assigned to your device’s interface.

statistics

get pppoe statistics

statistics — Specifies the statistics information.

update-dhcpserver

set pppoe update-dhcpserver
unset pppoe update-dhcpserver

update-dhcpserver — Specifies that the DHCP server (on the device) automatically updates DNS parameters received through the PPPoE connection.

user-name

set pppoe username *name_str* password *pswd_str*

username	Sets the username and password.

Example: The following command sets the username to Phred and Phred's password to !@%)&&:

```
set pppoe username Phred password !@%)&&
```

Defaults

The defaults for this command are as follows:

- Feature *disabled*
- Authentication method *any*
- Timeout *30* minutes
- auto-connect setting *disabled*
- lcp-echo-timeout value *180* seconds
- retries value *10*
- netmask value *255.255.255.255*
- update-dhcpserver setting *enabled*
- static-ip setting *disabled*
- clear-on-disconnect setting *disabled*
- ipcp ipv6cp switches assumed. The device attempts IPCP first, then IPv6CP.

proxy-id

Use the **proxy-id** commands to set device behavior for processing proxy ID updates. A proxy ID is a three-part tuple consisting of local IP address, remote IP address, and service. The proxy ID for both peers must match, which means that the service specified in the proxy ID for both peers must be the same, and the local IP address specified for one peer must be the same as the remote IP address specified for the other peer. The peers exchange proxy IDs during IKE Phase 2 negotiations.

During the startup process, the security device loads its configuration file. While loading this file, the security device reads the policies before the routes. Because of this, routing information that involves MIPs or VIPs can result in the security device deriving incorrect proxy-IDs from the policy information in the file. To resolve this problem, you can use the **unset proxy-id manual-update** command to change the default behavior of the device to update proxy IDs after the configuration file finishes loading. However, if you have a large number of policies, the update procedure can take a very long time to complete. By default, the device behavior does not update proxy IDs automatically during startup. Instead, you must manually update proxy IDs by entering the **exec proxy-id update** command. For VPN traffic that uses source or destination address translation, we recommend either of the following approaches:

- Use routing-based VPNs and separate the VPN and its manually defined proxy ID from the policy that enforces address translation.
- Use policy-based VPNs and assign proxy IDs to the VPN tunnels referenced by the policies rather than allow the security device to automatically derive the proxy IDs from the policies.

Syntax

exec

```
exec proxy-id update
```

get

```
get proxy-id
```

set

```
set proxy-id manual-update
```

unset

```
unset proxy-id manual-update
```

Keywords and Variables

update

```
exec proxy-id update
```

update	Instructs the security device to update all VPN proxy IDs.

manual-update

```
set proxy-id manual-update
unset proxy-id manual-update
```

manual-update	When set, instructs the security device to update all VPN proxy IDs only in response to the **exec proxy-id update** command. When unset, instructs the security device to update the proxy IDs automatically during route change.

Defaults

By default, the security device does not update proxy IDs automatically.

reset

Use the **reset** command to restart the NetScreen device.

Syntax

```
reset
    [
    no-prompt |
    save-config { no | yes } [ no-prompt ]
    ]
```

Keywords and Variables

no-prompt

```
reset no-prompt
```

no-prompt	Indicates no confirmation.

save-config

```
reset save-config [ no | yes ] [ no-prompt ]
```

save-config	■ **no** Directs the security device to not save the current configuration before resetting. ■ **yes** Directs the security device to save the current configuration before resetting. ■ **no-prompt** Does not display a confirmation prompt.

RIPng Commands

Use the ripng context to begin configuring Routing Information Protocol Next Generation (RIPng) for a Juniper Networks virtual router.

Initiating the ripng context can take up to four steps.

1. Enter the vrouter context by executing the set vrouter command.

   ```
   set vrouter vrouter
   ```

   For example:

   ```
   set vrouter trust-vr
   ```

2. Enter the ripng context by executing the set protocol ripng command.

   ```
   device(trust-vr)-> set protocol ripng
   ```

3. Enable RIPng (it is disabled by default).

   ```
   device(trust-vr/ripng)-> set enable
   ```

Table 2 shows the commands that are executable in the ripng context.

Table 2: RIPng Commands (Sheet 1 of 2)

Command	RIPng Function
advertise-def-route	Use the advertise-def-route commands to advertise the default route (0.0.0.0/0) of the current virtual router the RIPng routing domain. Every virtual router may have a default route entry, which matches every destination. (Any entry with a more specific prefix overrides the default route entry.) Command options: get, set, unset
config	Use the config command to view all of the commands executed to configure the RIPng routing instance. Command option: get
debug	Use the debug command to view detailed information about the RIPng instance. Command option: get
default-metric	Use the default-metric commands to set the RIPng metric for redistributed routes. The default value is 10. Command options: set, unset
enable	Use the enable commands to enable or disable RIPng in the virtual router. Command options: set, unset
flush-timer	Use the flush-timer commands to configure the number of seconds that elapse before ScreenOS automatically removes an invalidated route. The default is 120 seconds. Command options: set, unset
garbage-list	Displays all routes currently contained in the RIPng garbage list. This list contains routes automatically removed from the routing table because the device did not obtain the routes in the time interval specified by the Invalid Timer setting. When the Flush Timer interval elapses for an entry, the device purges the entry from the garbage list. Command option: get

Table 2: RIPng Commands (Sheet 2 of 2)

Command	RIPng Function
interface	Use the interface command to display all RIPng interfaces in the virtual router. Command options: get
invalid-timer	Use the invalid-timer commands to configure the number of seconds that elapse after a neighbor stops advertising a route before the route becomes invalid. The default is 180 seconds. Command options: set, unset
max-neighbor-count	Use the max-neighbor-count commands to set the maximum number of RIPng neighbors allowed. The default is 16. Command options: set, unset
neighbors	Use the neighbors command to display the status of RIPng neighbors. Command options: get
redistribute	Use the redistribute commands to import known routes from a router running a different protocol into the current routing instance. You can import the following types of routes: ■ Manually created (static) routes ■ BGP routes ■ OSPF routes ■ Routes created by an external router, due to reconnection of an interface with an IP address ■ Routes imported from other virtual routes Command options: set, unset
reject-default-route	Use the reject-default-route commands to cause RIPng to reject a default route learned from another protocol. Command options: get, set, unset
route-map	Use the route-map commands to filter routes and offset the metric to a RIPng route matrix. Command options: get, set, unset
routes-redistribute	Use the routes-redistribute command to display redistributed routes. Command options: get
rules-redistribute	Use the rules-redistribute command to display redistribution rules. Command options: get
threshold-update	Use the threshold-update commands to set the maximum number of routing packets allowed per update interval. Command options: set, unset
timer	Use the timer command to display RIPng timers. Command options: get
trusted-neighbors	Use the trusted-neighbors commands to set an access list that defines RIPng neighbors. Command options: get, set, unset
update-timer	Use the update-timer commands to set the interval, in seconds, when route updates are issued to RIPng neighbors. Command options: set, unset
update-threshold	Use the update-threshold command to display the number of routing packets per update interval. Command options: get

advertise-def-route

Use the **advertise-def-route** commands to advertise the default route (0::/0) of the current virtual router.

Every router might have a default route entry, which matches every destination. (Any entry with a more specific prefix overrides the default route entry.)

Before you can execute the **advertise-def-route** commands, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

get

```
get advertise-def-route
```

set

```
set advertise-def-route [ always ] {metric number | preserve-metric }
```

unset

```
unset advertise-def-route
```

Keywords and Variables

always

```
set advertise-def-route always ...
```

always	Directs the routing instance to advertise the default route under all conditions, even if there is no default route in the routing table. If you specify **always**, you must also specify the **metric** parameter; you can optionally specify the **preserve-metric** parameter. If you do not specify **always**, you must specify either the **metric** or **preserve-metric** option.

metric

```
set advertise-def-route always metric number
```

metric	Specifies the metric (cost), which indicates the overhead associated with the default route. Enter a number between 1 and 15. You must specify this parameter if you specify the **always** option.

preserve-metric

```
set advertise-def-route ... [ preserve-metric ]
```

preserve-metric	Instructs the virtual router to use the original (source) route's metric for advertisement when the route is redistributed. When you execute a **preserve-metric** command, in conjunction with a value specified by the **metric** command, the **preserve-metric** parameter takes precedence over the metric value when a route is redistributed.

config

Use the **config** command to display all commands executed to configure the RIPng local virtual router.

Before you can execute the **config** command, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

```
get config
```

Keywords and Variables

None.

debug

Use the **debug** command to view the RIPng debug information.

Syntax

```
get debug
```

Keywords and Variables

None.

default-metric

Use the **default-metric** commands to set the RIPng metric for redistributed routes.

Before you can execute the **default-metric** commands, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

set default-metric *number*

Keywords and Variables

Variables

set default-metric *number*

number	The metric for the routes redistributed into RIPng. This metric value can be from 1 to 15.

enable

Use the **enable** commands to enable or disable RIPng from the current virtual router.

Before you can execute the **enable** commands, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

```
set enable
```

Keywords and Variables

None.

flush-timer

Use the **flush-timer** commands to configure the time that elapses before an invalid route is removed.

Before you can execute the **flush-timer** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

```
set flush-timer number
```

Keywords and Variables

Variables

```
set flush-timer number
```

number	The number of seconds that elapses before an invalid route is removed. This value must be greater than the current **update-timer** value. The default value is 120 seconds.

garbage-list

Use the **garbage-list** commands to display all routes currently contained in the RIPng garbage list. The garbage list contains routes automatically removed from the routing table because the device did not obtain the routes in the time interval specified by the Invalid Timer setting. When the Flush Timer interval elapses for an entry, the device automatically purges the entry from the garbage list.

Before you can execute the **garbage-list** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

```
get garbage-list
```

Keywords and Variables

None.

interface

Use the **interface** command to display all RIPng interfaces on the current virtual router.

Before you can execute the **interface** command, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

```
get interface
```

Keywords and Variables

None.

invalid-timer

Use the **invalid-timer** commands to configure the time that elapses after a neighbor stops advertising a route before the route becomes invalid.

Before you can execute the **invalid-timer** commands, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

```
set invalid-timer number
```

Keywords and Variables

Variables

```
set invalid-timer number
```

number	The number of seconds after a neighbor stops advertising a route that the route becomes invalid. This value must be greater than the current **update-timer** value. The default value is 180 seconds.

max-neighbor-count

Use the **max-neighbor-count** commands to set the maximum number of RIPng neighbors allowed.

Before you can execute the **max-neighbor-count** commands, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

```
set max-neighbor-count number
```

Keywords and Variables

Variables

```
set max-neighbor-count number
```

number	The maximum number of RIPng neighbors allowed. This value can be from one to the maximum value possible for your security device. The default is 16 RIPng neighbors.

neighbors

Use the **neighbors** command to display the status of all RIPng neighbors.

Before you can execute the **neighbors** command, you must initiate the **ripng** context. (See "Initiating the ripng context can take up to four steps." on page 217.)

Syntax

```
get neighbors
```

Keywords and Variables

None.

redistribute

Use the **redistribute** commands to import known routes from a router running a different protocol into the current RIPng routing instance.

You can import the following types of routes:

- Manually created routes (**static**)
- BGP routes (**bgp**)
- OSPF routes (**ospf**)
- Directly connected interface with an IP address assigned to it (**connected**)
- Routes that have already been imported (**imported**)

Before you can execute the **redistribute** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get

get routes-redistribute

set

set redistribute route-map *name_str* protocol
{ bgp | connected | imported | ospf | static }

Keywords and Variables

protocol

set redistribute route-map *name_str* protocol { ... }

protocol	Specifies the routing protocol type. The route map can use the protocol type to the determine whether to permit or deny a route. ■ **bgp** specifies that the route map performs an action only on BGP routes in the subnetwork. ■ **connected** specifies that the route map performs an action only on routes sent from an external router that has at least one interface with an IP address assigned to it. ■ **imported** specifies that the route map performs an action only on imported routes in the subnetwork. ■ **ospf** specifies that the route map performs an action only on OSPF routes in the subnetwork. ■ **static** specifies that the route map performs an action only on static routes in the subnetwork.

route-map

set redistribute route-map *name_str* protocol { ... }

route-map	Identifies the route map that specifies the routes to be imported.

Example: The following command redistributes a route that originated from a BGP routing domain into the current RIPng routing instance:

```
device(trust-vr/ripng)-> set redistribute route-map map1 protocol bgp
```

reject-default-route

Use the **reject-default-route** commands to cause RIPng to reject default routes learned from a neighbor in the RIPng domain.

Before you can execute the **reject-default-route** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get

```
get reject-default-route
```

set

```
set reject-default-route
```

Keywords and Variables

None.

route-map

Use the **route-map** commands to filter incoming or outgoing routes.

Before you can execute the **route-map** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get

```
get route-map
```

set

set route-map *name_str* { in | out }

Keywords and Variables

Variables

set route-map *name_str*

name_str	The name of the route map to filter routes.

in

set route-map *name_str* in

in	Specifies the route map is applied to routes to be learned by RIPng.

out

set route-map *name_str* out

out	Specifies the route map is applied to routes to be advertised by RIPng.

Example: The following command applies the route map map1 to routes to be advertised by RIPng:

```
device(trust-vr/ripng)-> set route-map map1 out
```

routes-redistribute

Use the routes-redistribute command to display details about routes imported from other protocols into RIPng.

Before you can execute the routes-redistribute command, you must initiate the ripng context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

```
get routes-redistribute
```

Keywords and Variables

None.

rules-redistribute

Use the rules-redistribute command to display conditions set for routes imported from other protocols into RIPng.

Before you can execute the rules-redistribute command, you must initiate the ripng context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

```
get rules-redistribute
```

Keywords and Variables

None.

threshold-update

Use the threshold-update commands to set the maximum number of routing packets received and processed per update interval, per neighbor.

Before you can execute the threshold-update commands, you must initiate the ripng context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

set threshold-update *number*

Keywords and Variables

Variables

set threshold-update *number*

number	The maximum number of routing packets allowed per update interval. This value must be greater than zero.

timer

Use the **timer** command to display information about various RIPng timers.

Before you can execute the **timer** command, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get timer

Keywords and Variables

None.

trusted-neighbors

Use the **trusted-neighbors** commands to specify an access list that defines allowed RIPng neighbors.

Before you can execute the **trusted-neighbors** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get

get trusted-neighbors

set

set trusted-neighbors *id_num*

Keywords and Variables

Variables

set trusted-neighbors *id_num*

id_num	The number of the access list that defines the allowed RIPng neighbors.

update-timer

Use the **update-timer** commands to set the interval that RIPng sends route updates to neighbors.

Before you can execute the **update-timer** commands, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

set update-timer *number*

Keywords and Variables

Variables

set update-timer *number*

number	The interval, in seconds, that RIPng sends route updates to neighbors. This value must be at least one, and no greater than the current **invalid-timer** value. The default is 30 seconds.

update-threshold

Use the **update-threshold** command to display the number of routing packets per update interval.

Before you can execute the **update-threshold** command, you must initiate the **ripng** context. (See “Initiating the ripng context can take up to four steps.” on page 217.)

Syntax

get update-threshold

Keywords and Variables

None.

Chapter 4

rm Through zone

This section lists and describes the ScreenOS command line interface (CLI) commands **rm** through **zone**.

NOTE: Certain commands and features are platform-dependent and might be unavailable on your Juniper Networks security product platform. Check your product datasheet for feature availability.

Click on a command for details:

rm
route
sa
sa-filter
sa-statistics
save
scheduler
scp
service
session
snmp
socket
syslog
system
tech-support
tftp
timer
trace-route
traffic-shaping
url
user
user-group
vip
vlan
vpn
vpn-group
vpnmonitor
vrouter
vsys
vsys-profile
webauth
webtrends
xauth
xlate
zone

rm

Use the **rm** commands to view the resource manager settings. These settings include client heatbeat and pinhole age.

Syntax

```
get rm
      {
      group { id_number | active } |
      resource { id_number | active } |
      settings
      }
```

Keywords and Variables

rm	Displays resource manager details. The arguments for this keyword are as follows: ■ **group** { *id_number* | **active** } displays information about a particular group. You can limit the output by appending a specific group ID number or expand the output by appending **active** include all active groups in the output. ■ **resource** { *id_number* | **active** } displays information about a particular resource. You can limit the output by appending a specific resource ID number or expand the output by appending **active** to include all active groups in the output. ■ **settings** displays the client heatbeat (timeout and count) and pinhole age (in seconds).

route

Use the **route** commands to display entries in the static route table.

The **get route** command displays:

- The IP address, netmask, interface, gateway, protocol, preference, metric, and owner vsys
- The **protocol** value can be any of the following:
  - C (Connected)
  - S (Static)
  - A (Auto Exported)
  - I (Imported; that is, route imported from another virtual router)
  - iB (internal BGP)
  - eB (external BGP)
  - O (OSPF)
  - E1 (OSPF external type 1)
  - E2 (OSPF external type 2)

Use the **get route** command to see if the security device has a route to the IP address on the correct interface.

Syntax

```
get route [ v4 | v6 ]
    [
    id id_num |
    ip [ ip_addr ] |
    prefix ip_addr/mask |
    protocol { bgp | connected | imported | ospf | rip | static } |
    summary
    ]
```

Keywords and Variables

id

get route [v4 | v6] id *id_num*

id	Displays information about a specific route, identified by the ID number *id_num*.

Example 1: The following command displays the route information for any route (IPv4 or IPv6) with ID number 47:

get route id 47

Example 2: The following command displays the route information for an IPv6 route with ID number 35:

get route v6 id 35

ip

get route ip *ip_addr*

ip	Displays information about a specific route, identified by the destination IPv4 or IPv6 address (*ip_addr*).

Example: The following command displays the route information to a machine with the IPv6 address a172:ef16:60a3::1

get route ip a172:ef16:60a3::1

prefix

get route prefix *ip_addr*/*mask*

prefix	Displays routes within a specified subnet (*ip_addr*/*mask* or *ip_addr/prefix*).

Example 1: The following command displays the routes within the subnet 1.1.1.1/24:

get route prefix 1.1.1.1/24

Example 2: The following command displays the routes within the IPv6 prefix 3f45:a344:aa88::/48:

get route prefix 3f45:a344:aa88::/48

protocol

get route protocol { bgp **|** connected **|** discovered **|** imported **|** ospf **|** prefix **|** rip **|** ripng **|** static }

protocol

Specifies the routing protocol, and directs the security device to display the routes derived from that protocol.

- bgp Directs the device to display BGP routes.
- connected Directs the device to display only routes sent from an external router that has at least one interface with an IP address assigned to it.
- discovered Directs the device to display only routes discovered from the subnetwork.
- imported Directs the device to display imported routes.
- ospf Directs the device to display only OSPF routes.
- prefix Directs the device to display routes that have assigned prefixes.
- rip Directs the device to display RIP routes (for IPv4 only).
- ripng Directs the device to display RIPng routes (for IPv6 only).
- static Directs the device to display only static routes.

source

get route source { id *id_num* **|** in-interface { ... }**|** ip *ip_addr* **|** prefix *ip_addr*/*pref_len* }

source

Displays summary information for source routes.

- id *id_num* Displays a source route, identified by *id_num*.
- in-interface Displays SIBR routes. You can choose to identify a particular route by including an ID number, an IPv4 address, or an IPv6 prefix.
- ip *ip_addr* Displays a source route, identified by an IPv4 or IPv6 address, *ip_addr*.
- prefix *ip_addr*/*pref_len* Displays all source routes within the prefix specified by *ip_addr*/*pref_len*.

summary

get route summary

summary

Displays summary information, including number of routes, for each protocol.

Defaults

The **get route** command displays all entries in the route table unless a particular target IP address is specified.

sa

Use the sa commands to display active or inactive security associations (SAs) or to clear a specified SA.

A *security association* (SA) is a unidirectional agreement between VPN participants regarding the methods and parameters to use while securing a communication channel. Full bidirectional communication requires at least two SAs, one for each direction.

An SA groups together the following components for securing communications:

- Security algorithms and keys
- Protocol mode (transport or tunnel)
- Key management method (Manual Key or AutoKey IKE)
- SA lifetime

For outbound VPN traffic, a security policy invokes the SA associated with the VPN tunnel. For inbound traffic, the security device looks up the SA by using the following triplet: destination IP, security protocol (AH or ESP), and security parameter index (SPI) value, which are sent to the peer in the first message of a Phase 1 IKE exchange.

Syntax

clear

```
clear [ cluster ] sa id_num
```

get

```
get sa
    [
    id id_num |
    [ active | inactive ] stat
    ]
```

Keywords and Variables

Variables

clear [cluster] sa *id_num*

id_num	Specifies the SA ID number.

cluster

clear cluster sa *id_num*

cluster	Propagates the **clear** operation to all other devices in an NSRP cluster.

id

get sa id *id_num*

id	Displays a specific IPSec Security Association (SA) entry with the ID number.

stat

get sa [...] stat

stat

Shows the SA statistics for the device.

Displays these statistics for all incoming or outgoing SA pairs:

- **Fragment:** The total number of fragmented incoming and outgoing packets.
- **Auth-fail:** The total number of packets for which authentication has failed.
- **Other:** The total number of miscellaneous internal error conditions other than those listed in the auth-fail category.
- **Total Bytes:** The amount of active incoming and outgoing traffic

sa-filter

Use the **sa-filter** command to debug messages for each Security Association (SA) filter.

Syntax

get

get sa-filter

set

set sa-filter *ip_addr*

Keywords and Variables

Variables

set sa-filter *ip_addr*
unset sa-filter *ip_addr*

ip_addr	Specifies an Internet Protocol address (IP Address) for the SA to filter.

all

unset sa-filter all

all	Unsets all SA filters.

sa-statistics

Use the **sa-statistics** command to clear all statistical information (such as the number of fragmentations and total bytes through the tunnel) in a security association (SA) for an AutoKey IKE VPN tunnel.

Syntax

clear [cluster] sa-statistics [id *id_num*]

Keywords and Variables

cluster

clear cluster sa-statistics id *id_num*

cluster	If the security device is in a high availability (HA) configuration, propagates the **clear** operation to all other devices in the NetScreen Redundancy Protocol (NSRP) cluster.

id

clear [cluster] sa-statistics id *id_num*

id *id_num*	Clears the SA statistics for a particular SA.

Example: The following command clears the SA statistics for SA 2:

clear sa-statistics id 2

save

Use the **save** commands to save the NetScreen device configuration settings either to flash memory, a flash memory card, or a Trivial File Transfer Protocol (TFTP) server.

Syntax

save

```
save
```

save config

```
save config
    [
    all-virtual-system |
    to
      {
      flash |
      slot1 filename |
      tftp ip_addr filename
      }
        [ merge ] |
    from
      {
      flash |
      slot1 filename |
      tftp ip_addr filename
        [
        merge |
        to
          {
          flash [ from interface ] |
          last-known-good |
          slot1 filename |
          tftp ip_addr filename [ from interface ]
          }
            [ from interface ]
      }
    ]
```

save image-key

```
    save image-key tftp ip_addr filename from interface
```

save software

```
save software from
    {
    flash |
    slot1 filename |
    tftp ip_addr filename
    }
      to
        {
        flash |
        slot1 filename |
        tftp ip_addr filename
        }
          [ from interface ]
```

Keywords and Variables

all-virtual-system

```
save config all-virtual-system
```

all-virtual-system	Saves all virtual system configurations.

flash

```
save config from { ... } to flash [ from interface ]
save config from flash to { ... } [ from interface ]
save software from flash to { ... } [ from interface ]
save software from { ... } flash to [ from interface ]
```

flash	Saves from (or to) flash memory. The from *interface* option specifies the source interface if you specify TFTP.

Example 1: The following command saves the current configuration from flash memory to a file (output.txt) on a TFTP server (172.16.10.10):

```
save config from flash to tftp 172.16.10.10 output.txt
```

Example 2: The following command saves the current configuration from flash memory to a file (output.txt) on an IPv6 TFTP server (2eee::1):

```
save config from flash to tftp 2eee::1 output.txt
```

from { ... } to

```
save config from { ... } to { ... }
save software from { ... } to { ... }
```

from	Saves from the specified source.
to	Saves to the specified destination.

Example 1: The following command saves the current configuration from flash memory to a file (output.txt) on a TFTP server (IP address 172.16.10.10):

```
save config from flash to tftp 172.16.10.10 output.txt
```

Example 2: The following command saves the current configuration from flash memory to a file (output.txt) on an IPV6 TFTP server (IP address 2eee::1):

```
save config from flash to tftp 2eee::1 output.txt
```

last-known-good

save config to last-known-good

last-known-good	Saves the current configuration to flash memory as the LKG (last-known-good) configuration. The security device can revert to this LKG file by doing a configuration rollback. The security device automatically names the LKG file *$LKG$.cfg*. You cannot rename the LKG file or give it a different name upon saving it.

merge

save config from { ... } merge [from *interface*]

merge	Merges the saved configuration with the current configuration. The from *interface* option specifies the source interface.

Example: The following command merges the current configuration with the configuration in a file (input.txt) on a TFTP server (IP address 172.16.10.10):

```
save config from tftp 172.16.10.10 input.txt merge
```

Example 2: The following command merges the current configuration with the configuration in a file (input.txt) on an IPv6 TFTP server (IP address 2eee::1):

```
save config from tftp 2eee::1 input.txt merge
```

slot1

save config from { ... } to slot1 [...]
save config from slot1 to { ... }
save software from slot1 to { ... }
save software from { ... } to slot1 [...]

slot1	Saves from (or to) a file in the memory card slot.

Example: The following commands saves the current configuration from a file (input.txt) in the slot1 memory card to flash memory:

```
save config from slot1 input.txt to flash
```

tftp

save config from tftp *filename* to { … } [from *interface*]
save image-key tftp *ip_addr filename* from *interface*
save software from tftp *filename* to { … } [from *interface*]

tftp	Saves from (or to) a file on a TFTP server.

Example 1: The following command loads an authentication key on a FIPS-compliant security device from a file named nskey.cer on a TFTP server at 10.10.1.2:

save image-key tftp 10.10.1.2 nskey.cer

Example 2: The following command loads an authentication key on a FIPS-compliant security device from a file named nskey.cer on an IPv6 TFTP server at 2eee::1:

save image-key tftp 2eee::1 nskey.cer

scheduler

Use the scheduler commands to create or modify a schedule, or to display the settings in a schedule.

A *schedule* is a configurable object that you can use to define when policies are in effect. security devices use schedules to enforce the policies at specified times or intervals. Through the application of schedules, you can control network traffic flow and enforce network security.

Syntax

get

get scheduler [name *name_str* | once | recurrent]

set

```
set scheduler name_str
    [
    once start date time stop date time [ comment string ] |
    recurrent
      { monday | tuesday | wednesday | thursday | friday | saturday | sunday
        start time stop time
      }
          [ start time stop time ]
            [ comment string ]
    ]
```

Keywords and Variables

name

get scheduler name *name_str*

name *name_str* Defines a name for the schedule.

once

get scheduler once
set scheduler *name_str* once start *date time* stop *date time* [...]

once Apply the schedule once, starting on the day, month, year, hour, and minute defined, and stopping on the month, day, year, hour, and minute defined.

recurrent

get scheduler recurrent
set scheduler *name_str* recurrent { ... } [...]

recurrent Directs the security device to repeat the schedule according to the defined day of the week, hour, and minutes.

- monday Repeats every Monday.
- tuesday Repeat every Tuesday.
- wednesday Repeat every Wednesday.
- thursday Repeat every Thursday.
- friday Repeat every Friday.
- saturday Repeat every Saturday.
- sunday Repeat every Sunday.
  - start Defines when to start the schedule.
  - stop Defines when to stop the schedule.
  - comment Defines a descriptive character string.

start | stop

set scheduler *name_str* once start *date time* stop *date time* [...]
set scheduler *name_str* recurrent { ... } start time *stop time* [...]

start | stop Defines the day, month, and year (*date*) in USA format (mm/dd/yyyy).

Defines the hour and minutes (*time*) in the 24-hour clock format (hh:mm).

Example 1: The following command creates a schedule definition named “mytime” which starts on 1/10/2003 at 11:00 AM and ends on 2/12/2003 at 7:00 PM:

set scheduler mytime once start 1/10/2003 11:00 stop 2/12/2003 19:00

Example 2: The following command creates a schedule definition named “weekend” which starts at 8:00 AM and ends at 5:00 PM and repeats every Saturday and Sunday:

set scheduler weekend recurrent saturday start 8:00 stop 17:00
set scheduler weekend recurrent sunday start 8:00 stop 17:00

scp

Use the **scp** commands to configure the Secure Copy (SCP) client/server on security devices. SCP provides a way of transferring files to or from the security device using the SSH protocol.

NOTE: It is possible to initiate file transfer from an external host, not from the security device itself.

Syntax

get

```
get scp
```

set

```
set scp enable
```

Keywords and Variables

enable

```
set scp enable
unset scp enable
```

enable	Enables the Secure Copy (SCP) task. When SCP is enabled, the SSH task is activated if it is not already active.

service

Use the **service** commands to create custom service definitions, modify existing service definitions, or display the current entries in the service definition list.

Use service definitions in policies to specify how the security device provides a service during a secure session. For example, a custom service definition might permit sessions using TCP protocol to exchange traffic between specified source and destination ports. Any policy that uses this definition conforms to these specifications.

Syntax

get

```
get service
    [
    name_str |
    group [ name_str ] |
    pre-defined |
    timeout { other | tcp | udp } [ port number1 [ number2 ] ] |
    user
    ]
```

set

```
set service name_str
    [
    +
      {
      icmp type number code number |
      ptcl_num | tcp | udp
        { src-port number-number dst-port number-number }
      } |
    protocol
      {
      ptcl_num | tcp | udp
        [ src-port number-number ]
          [ dst-port number-number [ timeout { number | never } ] ] |
      icmp type number code number |
      } |
    timeout { number | never }
    ]
```

Keywords and Variables

Variables

```
get service name_str
set service name_str [ … ]
unset service name_str
```

name_str	Defines a name for the service.

+

```
set service name_str + { … }
```

+	Appends a service entry to the custom services list.

pre-defined

```
get service pre-defined
```

pre-defined	Displays all the predefined services.

protocol

```
set service name_str protocol { … } [ … ]
```

protocol

Defines the service by IP protocol.

Defines a protocol for the specified service.

- *ptcl_num* specifies the protocol by protocol number.
- icmp specifies a ICMP-based service.
  - type identifies the ICMP message type, for example, "Destination Unreachable".
  - code identifies a specific message from a ICMP message type group. For example, from the Destination Unreachable type group, there are various more specific messages identified by code such as Net Unreachable, Host Unreachable, Protocol Unreachable, and so on.
- tcp specifies a TCP-based service.
- udp specifies a UDP-based service.

Example: The following command sets a service named "ipsec" that uses protocol 50:

set service ipsec protocol 50

src-port | dst-port

```
set service name_str protocol { ... }
    [ src-port number-number ] [ dst-port number-number ]
```

src-port

Defines a range of source port numbers valid for the service and protocol.

dst-port

Defines a range of destination port numbers valid for the service and protocol.

Example: The following command sets a service named "test1" that uses destination tcp port 1001:

set service test1 protocol tcp src-port 0-65535 dst-port 1001-1001

timeout

```
get service timeout { other | tcp | udp } [ port number1 [ number2 ] ]
set service name_str timeout { number | never }
unset service name_str timeout
```

timeout

Sets or displays the timeout value for session sessions created on a port for TCP, UDP, or other protocols. You can specify session timeout value (<number>) in minutes, or as never.

Example1: The following command sets telnet service to timeout after 10 minutes:

set service telnet timeout 10

Example2: The following command displays timeouts for UDP from port 1720 to 1800:

get service timeout udp port 1720 1800

user

get service user

user	Displays all user-defined services.

Defaults

The default timeout for TCP connections is 30 minutes.

The default timeout for UDP connections is 1 minute.

NOTE: The timeout value for TCP connections and UDP connections is 2160 minutes.

Using the **get service** command without any arguments displays all predefined, user-defined, and service-group information in the service book.

session

Use the **session** commands to clear or display entries in the session table of the security device.

The *session table* contains information about individual sessions between hosts that communicate through the security device. Because each session entry uniquely identifies two communicating hosts, it contains a unique combination of the following criteria:

- An individual IP address for the source host (no subnets with multiple addresses).
- An individual IP address for the destination host (no subnets with multiple addresses).
- An individual port number for the source host (not a range of ports).
- An individual port number for the destination host (not a range of ports).

Every time the security device initiates a new session, it creates a session entry and uses the information in the entry while processing subsequent traffic between the hosts.

The kind of session information listed by the **get session** command depends upon the security device platform. (For example, on any platform with a management module in slot 1, the **get session** command lists currently active sessions on that module.) Such sessions include management, log, and other administrative traffic. On any security device with one or more Secure Port Modules (SPMs), the **get session** command lists sessions that are active on the ASIC for each module. If a session crosses two ASICs, it counts as two sessions, one for each ASIC.

Syntax

clear

```
clear [ cluster ] session
    [
    all |
    id id_num |
    [ src-ip ip_addr [ netmask mask ] ]
      [ dst-ip ip_addr [ netmask mask ] ]
        [ src-mac mac_addr ] [ dst-mac mac_addr ]
          [ protocol ptcl_num [ ptcl_num ] ]
            [ src-port port_num [ port_num ] ]
              [ dst-port port_num [ port_num ] ]
                [ vsd-id id_num ]
    ]
```

get

```
get session
    [
    id id_num |
    fragment |
    [ tunnel ]
      [ src-ip ip_addr [ netmask mask ] ]
        [ dst-ip ip_addr [ netmask mask ] ]
          [ src-mac mac_addr ] [ dst-mac mac_addr ]
            [ protocol ptcl_num [ ptcl_num ] ]
              [ src-port port_num [ port_num ] ]
                [ dst-port port_num [ port_num ] ]
    ]
```

Keywords and Variables

all

```
clear [ cluster ] session all
```

all	Specifies all sessions.

cluster

```
clear cluster session [ ... ]
```

cluster	Propagates the clear operation to all other devices in an NSRP cluster.

id

```
clear [ cluster ] session id id_num
get session id id_num
```

id *id_num*	Identifies a specific session with Session Identification number *id_num*.

Example: The following command displays the session table entry for the session with ID 5116:

```
get session id 5116
```

src-ip | dst-ip

```
clear [ cluster ] session [ src-ip ip_addr [ netmask mask ] ]
    [ dst-ip ip_addr [ netmask mask ] ] [ ... ]
get session [ ... ] [ src-ip ip_addr [ netmask mask ] ]
    [ dst-ip ip_addr [ netmask mask ] ][ ... ]
```

src-ip *ip_addr* Identifies all sessions initiated by packets containing source IP address *ip_addr*. For example, *ip_addr* could be the source IP address in the first TCP SYN packet. For clear commands, IPv4 or IPv6 addresses are acceptable.

dst-ip *ip_addr* Identifies all sessions initiated by packets containing destination IP address *ip_addr*. For clear commands, IPv4 or IPv6 addresses are acceptable.

Example 1: The following command displays all the entries in the session table for a specific source IP address:

```
get session src-ip 172.16.10.92
```

Example 2: The following command clears all the entries in the session table for a specific source IPv6 address:

```
clear session src-ip 2eee::1
```

src-mac | dst-mac

```
clear [ cluster ] session [ ... ] [ dst-ip ip_addr [ netmask mask ] ]
    [ src-mac mac_addr ] [ dst-mac mac_addr ]
get session [ ... ] [ src-ip ip_addr [ netmask mask ] ]
    [ dst-ip ip_addr [ netmask mask ] ]
```

src-mac Identifies all sessions initiated by packets containing source MAC address *mac_addr*.

dst-mac Identifies all sessions initiated by packets containing destination MAC address *mac_addr*.

protocol

```
clear [ cluster ] session [ ... ] protocol ptcl_num [ ptcl_num ] [ ... ]
get session [ ... ] protocol ptcl_num [ ptcl_num ] [ ... ]
```

protocol Identifies all sessions that use protocol *ptcl_num*.

You can also specify any protocol within a range (*ptcl_num ptcl_num*).

src-port | dst-port

```
clear [ cluster ] session [ ... ] [ src-port port_num [ port_num ] ]
    [ dst-port port_num [ port_num ] ] [ ... ]
get session [ ... ] [ src-port port_num [ port_num ] ]
    [ dst-port port_num [ port_num ] ]
```

src-port	Identifies all sessions initiated by packets that contain the layer 4 source port *port_num* in the layer 4 protocol header. You can also specify any layer 4 destination port within a range (*port_num port_num*).
dst-port	Identifies all sessions initiated by packets that contain the layer 4 destination port *port_num* in the layer 4 protocol header. You can also specify any layer 4 destination port within a range (*port_num port_num*).

Example: The following command displays all the entries in the session table for protocol 5 and for source ports 2 through 5:

get session protocol 5 src-port 2 5

tunnel

get session tunnel [...]

tunnel	Directs the security device to display tunnel sessions.

vsd-id

clear [cluster] session [...] vsd-id *id_num*
get session [...] vsd-id *id_num*

vsd-id *id_num*	Identifies all sessions that belong the VSD group *id_num*.

Example: The following command clears all sessions belonging to VSD group 2001, and initiated from the host at IP address 172.16.10.12:

clear session src-ip 172.16.10.12 vsd-id 2001

snmp

Use the snmp commands to configure the security device for Simple Network Management Protocol (SNMP), to gather statistical information from the security device, and receive notification when significant events occur.

Syntax

get

```
get snmp [ auth-trap | community name_str | settings ]
```

set

```
set snmp
    {
    auth-trap enable |
    community name_str
      { read-only | read-write }
        [
        trap-off |
        trap-on [ traffic ] |
        version { any | v1 | v2c }
        ] |
    contact name_str |
    host comm_name ip_addr[/mask ]
      [
      src-interface interface |
      trap { v1 | v2c }
      ] |
    location string |
    name name_str |
    port { listen [ port_num ] | trap [ port_num ] } |
    }
```

Keywords and Variables

auth-trap enable

```
get snmp auth-trap
set snmp auth-trap enable
unset snmp auth-trap enable
```

auth-trap enable Enables Simple Network Management Protocol (SNMP) authentication traps.

community

```
get snmp community name_str
set snmp community name_str { ... }
unset snmp community name_str
```

community — Defines the name for the SNMP community. It supports maximum 3 communities in all products.

- **read-only** Defines the permission for the community as “read-only.”
- **read-write** Defines the permission for the community as “read-write.”
  - **trap-off** Disables SNMP traps for the community.
  - **trap-on** Enables SNMP traps for the community. The **traffic** switch includes traffic alarms as SNMP traps.

Examples: The following command configures a community named “public”.

- Allows hosts to read MIB data from the SNMP agent
- Enables SNMP traps for the community

```
set snmp community public read-only trap-on
```

The following command configures an SNMP host with IP address 10.20.25.30 for the community named “public”:

```
set snmp host public 10.20.25.30
```

contact

```
set snmp contact name_str
unset snmp contact
```

contact — Defines the system contact.

host

```
set snmp host comm_name ip_addr [/mask ] [ ... ]
unset snmp host comm_name ip_addr [ ... ]
```

host — Defines the community name string and the IP address of the SNMP management host. The *mask* value defines an SNMP community member as a subnet.

Note: When you define an SNMP community member as a subnet, that member can poll the security device but it cannot receive SNMP traps. To receive SNMP traps, the community member must be a single host.

Example: The following commands configure a community named “netscreen.”

- Specifies read and write permission
- Allows the security device to send traps to all hosts in the community
- Assigns the community to an SNMP host with IP address 10.40.40.15

```
set snmp community netscreen read-write trap-on
set snmp host netscreen 10.40.40.15
```

Example: The following command defines the subnet 10.5.1.0/24 as a member of the SNMP community named olympia:

```
set snmp host olympia 10.5.1.0/24
```

location

set snmp location *string*
unset snmp location

location	Defines the physical location of the system.

name

set snmp name *name_str*
unset snmp name

name	Defines the name of the system.

port

set snmp port { ... }
unset snmp port { ... }

port	Specifies the SNMP listen and trap port (listen | trap).

settings

get snmp settings

settings	Displays the name of the contact person, and the name and physical location of the NetScreen device.

src-interface

set snmp host *comm_name ip_addr*[/*mask*] src-interface *interface*
unset snmp host *comm_name ip_addr*[/*mask*] src-interface

src-interface	Specifies the source interface.

trap

set snmp host *comm_name* *ip_addr*[/*mask*] trap v1 **|** v2c

trap	If an SNMP community supports both SNMP versions (SNMPv1 (v1) and SNMPv2c (v2c), you must specify a trap version for each community member.

version

set snmp community { … } version { any **|** v1 **|** v2c}

version	When you create an SNMP community, you can specify whether the community supports SNMPv1 (v1), SNMPv2c (v2c), or both SNMP versions, as required by the SNMP management stations. For backward compatibility with earlier ScreenOS releases that only support SNMPv1, security devices support SNMPv1 by default.

socket

Use the socket commands to display socket information about a security device.

A *socket* is a software object that serves as a connection to a network protocol. A security device can send and receive TCP/IP or UDP traffic by opening a socket and reading and writing data to and from the socket.

Syntax

get socket [id *id_num*]

Keywords and Variables

id

get socket id *id_num*

id	Displays the information for an identified socket (*id_num*).

Example: The following command displays the information concerning socket 5:

get socket id 5

syslog

Use the syslog commands to configure the security device to send traffic and event messages to up to four syslog hosts or to display the current syslog configuration.

NOTE: The syslog host must be enabled before you can enable syslog.

Syntax

get

```
get syslog
```

set

```
set syslog
    {
    config { name_str | ip_addr }
      [
      facilities
        AUTH/SEC |
        local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7
          {
          AUTH/SEC |
          local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7
          } |
      log { all | event | traffic } |
      port number |
      transport tcp
      ] |
    enable |
    src-interface interface |
    }
```

Keywords and Variables

config

```
set syslog config { name_str | ip_addr } { ... }
unset syslog config [ ip_addr ]
```

config	Defines the configuration settings for the syslog utility. The { *name_str* | *ip_addr* } parameters define the name or the IP address of the syslog host device. You can define up to four syslog hosts. Specifying an IPv4 or IPv6 address with the unset syslog config command removes the configuration for the specified syslog host. Otherwise, this command removes the configuration for all syslog hosts.

enable

set syslog enable
unset syslog enable

enable	Enables the security device to send messages to the syslog host(s).

facilities

```
set syslog config { name_str | ip_addr }
    facilities { AUTH/SEC | local0 | local1 | local2 | local3 | local4 | local5 | local6 |
    local7}
    { AUTH/SEC | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 }
```

facilities	Defines the *security facility level* and the *regular facility level* for each syslog host that you specify. The security facility classifies and sends messages to the syslog host for security-related actions such as attacks. The regular facility classifies and sends messages for events unrelated to security, such as user logins and logouts, and system status reports.

Example 1: The following command sets the syslog host configuration to report all logs:

set syslog config 172.16.20.249 facilities local0 local1

Example 2: The following command sets the syslog host configuration to an IPv6 address to report all logs:

set syslog config 2eee::1 facilities local0 local1

log

```
set syslog config { name_str | ip_addr } log all | event | traffic
unset syslog config { name_str | ip_addr } log all | event | traffic
```

log	Directs the security device to send traffic log entries, event log entries or both traffic and event log entries to the syslog host, which can have an IPv4 or IPv6 address.

port

```
set syslog config { name_str | ip_addr } port port_num
unset syslog config { name_str | ip_addr } port
```

port	Defines the port number on the syslog host that receives the User Datagram Protocol (UDP) packets from the security device.

Example: The following command changes the syslog port number to 911:

set syslog config port 911

src-interface

set syslog config { *name_str* **|** *ip_addr* } src-interface *interface*
unset syslog config { *name_str* **|** *ip_addr* } src-interface

src-interface	Specifies the source interface.

transport tcp

set syslog config { *ip_addr* **|** *name_str* } transport tcp

transport tcp	Directs the device to use TCP protocol instead of UDP protocol.

Defaults

This feature is disabled by default. The default syslog port number is 514, and the default WebTrends port number is 514.

system

Use the get system command to display general system information.

The information displayed by the get system command includes:

- Descriptive indices of the ScreenOS operating system, including Serial Number, Control Number, Software Number, and image source file name.
- Descriptive indices of the Juniper Networks hardware platform, including hardware version, MAC address, and type.
- Chronological and timekeeping information.
- Current operational mode (transparent, NAT, or route)
- Configuration port and user IP.
- Interface settings.

Syntax

get system

Keywords and Variables

None.

tech-support

Use the **tech-support** command to display system information.

The information displayed by the **get tech-support** command is useful for troubleshooting the NetScreen device. Most of this information consists of the current authentication and routing settings.

Syntax

```
get tech-support
```

Keywords and Variables

None.

tftp

Use the **tftp** commands to specify the interface the device uses to communicate via TFTP sessions.

Syntax

get

get tftp *ip_addr filename*

set

unset tftp source-interface *ip_addr*

Keywords and Variables

Variables

get tftp *ip_addr filename*
set tftp source-interface *ip_addr*

ip_addr	Specifies the IP address of the TFTP interface.
filename	Specifies the name of the file to access with the TFTP service.

action

source-interface	Specifies the IP address of the interface through which the device communicates using TFTP.

timer

Use the **timer** commands to display timer settings, or to configure the security device to automatically execute management or diagnosis at a specified time.

All timer settings remain in the configuration script after the specified time has expired.

Syntax

get

```
get timer
```

set

set timer *date time* action reset

Keywords and Variables

Variables

set timer *date time* action reset
unset timer *id_num*

date	Specifies the date when the security device executes the defined action. Date is in *mm/dd/yyyy* format.
time	Specifies the time when the security device executes the defined action. Time is in *hh:mm* format.
id_num	Identifies a specific action by ID number in the list of timer settings (generated by the **set timer** command.)

action

set timer *date time* action reset

action	Defines the event that the command triggers at the given date and time.

reset

set timer *date time* action reset

reset	Resets the timer.

Example: The following command configures ScreenOS to reset at a given time and date:

set timer 1/31/2000 19:00 action reset

trace-route

Use the **trace-route** command to display the route to a host.

Syntax

```
trace-route string
    [
    from interface |
    hop number
      [
      from interface |
      time-out number [ from interface ]
      ]
```

]*Keywords and Variables*

from

trace-route *string* [from *interface* | hop *number* [from *interface* | time-out *number* [from *interface*]]]

from	Specifies the interface from which to initiate the trace-route. In route mode, *interface* is the IP address you set on the source interface; in transparent mode, *interface* is the default VLAN1 IP address. The interface must be active and it can not be a loopback, null, HA, or tunnel interface, and it can not be in the null zone or be a bgroup member. The interface must be a pure Layer2 or Layer3 interface.

Example: The following command performs a trace-route operation from interface ethernet0/0 on host www.juniperdomain.com:

```
trace-route www.juniperdoman.com from ethernet0/0
```

hop

trace-route *string* [from *interface* | hop *number* [from *interface* | time-out *number* [from *interface*]]
]

hop	The maximum number of trace route hops (*number*) to evaluate and display.

Example: The following command sets a hop count of 4 in a trace-route operation on ethernet0/0 on device 1.1.1.1:

```
trace-route 1.1.1.1 hop 4 from ethernet0/0
```

string

trace-route *string* [...]

string	The name or IP address of the host.

time-out

trace-route *string* [from *interface* | hop *number* [from *interface* | time-out *number* [from *interface*]]] time-out Specifies the amount of time in seconds (*number*) to elapse before abandoning the route trace.

Example: The following command sets a hop count of 4 in a trace-route operation on ethernet0/0 on device 1.1.1.1:

```
trace-route 1.1.1.1 hop 20 timeout 20
```

Example 2: The following command performs a trace-route operation.

- Evaluates and displays up to four route trace hops
- Sends the output to a host with IPv6 address 2eee::1
- Specifies a timeout value of 4 seconds

trace-route 2eee::1 hop 4 time-out 4

traffic-shaping

Use the **traffic-shaping** commands to determine the settings for the system with the traffic-shaping function, or to display information about traffic management device interfaces.

Traffic shaping is the allocation of the appropriate amount of network bandwidth to every user and application on an interface. The appropriate amount of bandwidth is defined as cost-effective carrying capacity at a guaranteed Quality of Service (QoS). You can use a security device to shape traffic by creating policies and by applying appropriate rate controls to each class of traffic going through the security device.

NOTE: You can only apply traffic shaping to policies for which the destination zone has a single bound interface.

Syntax

get

```
get traffic-shaping
    {
    interface [ interface ] |
    ip_precedence |
    mode
    }
```

set

```
set traffic-shaping
    {
    ip_precedence
      number1 number2 number3 number4 number5 number6 number7 number8 |
    mode { auto | off | on }
    }
```

Keywords and Variables

interface

```
get traffic-shaping interface [ interface ]
```

interface	Displays the traffic shaping info for an interface.

ip_precedence

```
get traffic-shaping ip_precedence
set traffic-shaping ip_precedence
    number1 number2 number3 number4 number5 number6 number7 number8
unset traffic-shaping mode ip_precedence
```

ip_precedence	Specifies the Priorities 0 through 7 for IP precedence (TOS) mapping. Each setting should be a single-digit value.

mode

```
get traffic-shaping mode
set traffic-shaping mode { auto | off | on }
unset traffic-shaping mode
```

mode	Defines the mode settings for the system with the traffic-shaping function. If you select auto, the system automatically determines the mode settings. If there is at least one policy in the system with traffic-shaping turned on, the system automatically sets the mode to on. If there is no such policy, the auto mode default setting is off.

Defaults

By default, the traffic shaping function is set up to automatic mode.

url

Use the url commands to enable or disable URL filtering for use in policies, and to configure and display URL filter settings.

NOTE: A Websense server provides the URL filtering.

Syntax

get

```
get url [ all | vsys-name vsys_name ]
```

set (root level)

```
set url
    {
    config { disable | enable } |
    fail-mode { block | permit } |
    message string |
    server { ip_addr | dom_name } port_num number |
    src-interface interface |
    type { NetScreen | Websense }
    }
```

set (vsys level)

```
set url
    {
    config { disable | enable } |
    use-root |
    use-vsys
    }
```

Keywords and Variables

config

```
set url config { disable | enable }
unset url config
```

config { disable | enable }	Disables or enables URL filtering at the device level for use in policies. By itself, enabling URL filtering at the device level does not activate it. You must enable URL filtering at the device and policy level to apply filtering to URL requests.

fail-mode

```
set url fail-mode { block | permit }
unset url fail-mode
```

fail-mode { block | permit }	If the connection between the security device and the Websense server is lost, the security device either blocks or permits all HTTP requests to which a policy requiring URL filtering applies. The default fail-mode behavior is to block HTTP requests.

message

set url message *string*
unset url message

message *string* Defines a custom message, 1-500 characters in length, to send to the client who is blocked from reaching a URL.

Example: The following command defines the URL blocking message “This site is blocked”:

set url message “This site is blocked.”

server

set url server { *ip_addr* | *dom_name* } port_num *number*
unset url server

server Defines the following connection parameters for the URL filtering server:

- *ip_addr* | *dom_name* Sets the IPv4 or IPv6 address or DNS name of the URL filtering server.
- *port_num* Sets the port number on which the security device communicates with the URL filtering server. The default port number is 15868.
- *number* Sets the timeout interval, in seconds, that the security device waits for a response from the Websense filter. If Websense does not respond within the time interval, the security device either blocks the request or allows it, as you choose. The default is 10 seconds.

Example: The following command sets the IP address, port number, and timeout value for the URL filtering server (the port number and timeout interval use the default values):

set url server 1.2.2.20 15868 10

src-interface

set url src-interface *interface*
unset url src-interface

src-interface *interface* Specifies the source interface that the security device uses when communicating with the Websense server. If you specify a source interface, the device enforces use of that interface, without consulting the routing table. If you do not specify an interface, the device picks an interface according to entries in the routing table.

type

```
set url type { NetScreen | Websense }
unset url type
```

type { NetScreen | Websense }	Specifies the source of the message that the security device delivers to clients when URLs are blocked—the security device or the Websense server.

use-root

```
set url use-root
```

use-root	When entering this command in a virtual system (vsys), it instructs the vsys to share the URL filtering server defined at the root level.

use-vsys

```
set url use-vsys
```

use-vsys	When entering this command in a virtual system (vsys), it instructs the vsys to use the URL filtering server defined for that vsys.

user

Use the user commands to create, remove, or display entries in the internal user authentication database.

The basic user categories are as follows:

- Authentication users (for using network connections)
- IKE users (for using AutoKey IKE VPNs)
- L2TP users (for using L2TP tunnels)
- XAuth users

Syntax

get

```
get user { name_str | all | id id_num }
```

set

```
set user name_str
    {
    disable |
    enable |
    hash-password string |
    ike-id
      {
      asn1-dn { [ container string ] wildcard string } [ share-limit number ] |
      fqdn name_str |
      ip ip_addr |
      u-fqdn name_str
      } |
    password pswd_str |
    remote-settings
      {
      dns1 ip_addr |
      dns2 ip_addr |
      ipaddr ip_addr |
      ippool name_str |
      wins1 ip_addr |
      wins2 ip_addr
      } |
    type { [ auth ] [ ike ] [ l2tp ] [ xauth ] } |
    uid id_num
    }
```

Keywords and Variables

Variables

```
get user name_str
set user name_str { ... }
unset user name_str [ ... ]
```

user Defines the user’s name.

Example: The following command displays information for a user named “roger”:

get user roger

all

get user all

all	Displays the following information for all the entries in the internal user database: ■ User ID number ■ Username ■ Status (enabled or disabled) ■ User type ■ IKE ID types – email address, IP address, or domain name—and IKE identity

disable | enable

set user *name_str* disable
set user *name_str* enable

disable | enable	Disables or enables the user in the internal database. By default, the user is disabled. I you set a password for an auth user or an IKE ID for an IKE user, the user becomes enabled automatically.

id

get user id *id_num*

id	Displays information about the user, identified by id_num. This option displays the same information as get user *name_str* option.

Example: The following command displays a particular user with user ID “10”:

get user id 10

hash-password

set user *name_str* hash-password *string*

hash-password	Creates a hashed password for the specified user and stores it in the configuration. Only an auth user can have a hashed password. The security device generates a hashed password randomly using either the crypt () or SHA-1 algorithm.

ike-id

set user *name_str* ike-id { ... }

ike-id { ip_addr |name_str }	Adds and defines an AutoKey IKE dialup user. ■ ip *ip_addr* The IP address of the dialup user. ■ fqdn *name_str* The fully qualified domain name, the complete string, such as www.juniper.net. ■ u-fqdn *name_str* Specifies the dialup user identity, usually equivalent to an email address such as admin@acme.com. ■ asn1-dn Specifies the user certificate distinguished name fields, and field values that define user identity. ■ container *string* Specifies a container identity. This identity allows multiple identity fields for each type (CN, OU, O, L, ST, C, and E). To match a local ASN1_DN identity, the peer IKE identity fields must match all identity fields specified in the container identity. The security device does not check any undefined container fields. Field sequence must be identical. ■ wildcard *string* Specifies a wildcard identity. This identity allows only one identity field for each type (CN, OU, O, L, ST, C, and E). To match a local ASN1_DN identity configuration, the peer IKE identity must contain fields matching all non-empty identity fields specified in the wildcard identity. For example, the wildcard identity o = ACME,ou = Marketing allows tunnel communication with any user whose certificate contains these field values. The security device does not check any undefined wildcard fields. Field sequence is not important. ■ share-limit *number* Specifies the number of users that can establish tunnels concurrently using this identity. When this number is larger than 1, the security device treats it as a Group IKE ID user. With Group IKE ID, multiple dialup users can establish tunnels using partial IKE identities.

Example 1: The following command creates an IKE user named branchsf with the IKE-ID number 2.2.2.2:

set user branchsf ike-id ip 2.2.2.2

Example 2: The following command creates a new user definition named “market”.

- Configures the user definition to recognize up to 10 hosts
- Specifies that the hosts must possess certificates containing “ACME” in the O field, and “Marketing” in the OU field

  set user market ike-id asn1-dn wildcard “o=ACME,ou=Marketing” share-limit 10

(This command uses Group IKE ID, which allows multiple hosts to use a single user definition. For more information about Group IKE ID, refer to the *Concepts & Examples ScreenOS Reference Guide.*)

password

set user *name_str* password *pswd_str*

password	Defines a top-level password, used to authenticate the auth, L2TP, IKE or XAuth user.

Example: The following command creates an authentication user in the ScreenOS internal database for user guest with the password JnPc3g12:

set user guest password JnPc3g12

remote-settings

set user *name_str* remote-settings { dns1 *ip_addr* **|** dns2 *ip_addr* }
set user *name_str* remote-settings ipaddr *ip_addr*
set user *name_str* remote-settings ippool *name_str* **|**
set user *name_str* remote-settings { wins1 *ip_addr* **|** wins2 *ip_addr* }

remote-settings — Sets the remote settings for the user.

- dns1 | dns2 Specifies the IP address of the primary and secondary DNS servers.
- ipaddr Specifies the static IP address for the user.
- ippool Specifies the named L2TP IP pool, which contains a range of IP addresses. The security device uses IP pools when it assigns addresses to dialup users using L2TP. (To define a L2TP pool, use the set ippool command.)
- wins1 | wins2 Specifies primary and secondary servers that provide WINS (Windows Internet Naming Service). WINS is a service for mapping IP addresses to NetBIOS computer names on Windows NT server-based networks. A WINS server maps a NetBIOS name used in a Windows network environment to an IP address used on an IP-based network.

Example: The following command directs the device to obtain an IP address from an L2TP ippool named NY_Pool for a dialup user named John_Doe:

set user John_Doe remote-settings ippool NY_Pool

type

set user *name_str* type { [auth] [ike] [l2tp] [xauth] }

type	Sets the user type, in any of the following combinations: auth, ike, l2tp, xauth, auth ike l2tp xauth, auth ike, auth l2tp, auth xauth, ike l2tp, ike xauth, l2tp xauth, auth ike l2tp, auth l2tp xauth, or ike l2tp xauth.

Example: The following command changes the user guest to an authentication/L2TP user:

set user guest type auth l2tp

user-group

Use the user-group commands to create or delete a user group, to modify it, or to add or remove a user from it.

User groups allow policies to treat multiple users in the same way, thus avoiding individual configurations for individual users. For example, even though you can configure dialup VPN tunnels for IKE users on a per-user basis, it is often more efficient to aggregate the users into a group, for which only one tunnel configuration is necessary.

Any policy that references a user group applies to all the members in the group. An authentication user can be a member of up to four different user groups.

NOTE: Different Juniper Networks platforms allow a different number of members in a user group.

Syntax

get

get user-group { *name_str* **|** all **|** external **|** id *id_num* **|** local }

set

```
set user-group name_str
    {
    id id_num |
    location { external | local } |
    type { [ auth ] [ ike ] [ l2tp ] [ xauth ] } |
    user name_str
    }
```

Keywords and Variables

Variables

get user-group *name_str*
set user-group *name_str* { ... }
unset user-group *name_str* [...]

name_str	Specifies the name of the user group.

Example: The following command displays the contents of a user group named Corp_Dial:

get user-group Corp_Dial

all

get user-group all

all	Displays all existing user groups.

external

get user-group external
set user-group *name_str* location external

external	Defines a user group as external. You can store user definitions in groups on an external RADIUS server. You can then define a user group on the security device, define the type of user it contains, leave it unpopulated of users, and define the user group as external. Defining an external user group on the security device allows you to reference that group in policies requiring authentication. When the policy requires an authentication check, the security device then contacts the RADIUS server, which performs the authentication check.

id

get user-group id *id_num*
set user-group *name_str* id *id_num*
unset user-group *name_str* [...]

id	Identifies the user group with an identification number *id_num*.

Example: The following command creates a user group named Corp_Dial, and assigns the group an ID of 10:

set user-group Corp_Dial id 10

local

get user-group local

local	Displays all local user groups.

location

set user-group *name_str* location { external | local }
unset user-group *name_str* location

location

Specifies the location of the user group:

- external Indicates that the user group is stored on an external authentication server. (ScreenOS supports user groups on RADIUS servers.)
- local Indicates that the user group is stored in the local database on the security device.

type

set user-group *name_str* type { ... }

type	Specifies the type of user group when that group is stored on an external RADIUS server. (When the user-group is stored in the local database, the user types determine the type of user group.) The following are the possible user group types: ■ auth Specifies that the group is comprised of authentication users. ■ ike Specifies that the group uses IKE. ■ l2tp Specifies L2TP users. ■ xauth Specifies XAuth users.

user

set user-group *name_str* user *name_str*
unset user-group *name_str* user *name_str*

user *name_str*	Adds or removes the named user to the specified user group.

Example 1: The following commands create a new authentication user named guest:

- An authentication user group named Corp_Dial with ID 1010
- User added to the user group

```
set user guest password JnPc3g12
set user-group Corp_Dial location local
set user-group Corp_Dial user guest
```

Example 2: The following command removes the user guest from the group:

```
unset user-group Corp_Dial user guest
```

vip

Use the **vip** commands to display the virtual IP (VIP) address configuration settings and to enable all VIPs to support multi-port services.

A *virtual IP* (VIP) address maps traffic received at one IP address to another address based on the destination port number in the TCP or UDP segment header.

Syntax

get

```
get vip
      [
      ip_addr { port port_num | port-status } |
      server |
      session timeout
      ]
```

set

```
set vip
      {
      multi-port |
      session timeout number
      }
```

Keywords and Variables

Variables

get vip *ip_addr* { port *port_num* | port-status }

ip_addr	Identifies the VIP address.
port *port_num*	Identifies the destination port, so that the security device can display information about the specified virtual port defined on the VIP.
port-status	Displays information about port allocation on the specified VIP.

multi-port

set vip multi-port

multi-port	Enables the support of multiple virtual ports per custom service. By default, VIPs support single-port services. **Warning:** After you execute this command, you must restart the device. This command changes the functionality of the VIP, so it is highly inadvisable to switch back and forth between enabling and disabling the **multi-port** modes.

server

get vip server

server Displays the connectivity status of servers receiving traffic via VIPs.

session

get vip session

session timeout Displays the outstanding session timeout value for VIP.

vlan

Use the **vlan** commands to create and configure virtual local area networks (VLANs) and create VLAN retagging pairs. You can create a VLAN both at the root and under an already configured virtual system (vsys).

Syntax

get

```
get vlan
    [
    all |
    group [ all | string ] |
    import [ all ] |
    port [ all ] |
    retag [all | name [ string ] ]
    ]
```

set

```
set vlan
    {
    group { name name_str } |
    group name_str { vsd-group id number } |
    port interface
        group name_str { zone name_str } |
        retag
    retag { name name_str number number }
    }
```

Variable Parameters

all

get vlan all

all Displays information about all root level VLANs when run from the root and all vsys VLANs when run in a vsys.

group

get vlan group [**all** | *string*]
set vlan group { name *name_str* }
set vlan group *name_str* { vsd-group id *number* }
set vlan group *name_str* { vlan_low, vlan_high }
unset vlan group { *string* }
unset vlan group *name_str* { vsd-group id *number* }

group	Specifies an existing VLAN group or a VLAN group that you want to create. ■ **vsd-group id** specifies a collection of VLANs grouped together. You can select or assign any value from 2 to 4094.

import

get vlan import [all]
set vlan import { vlan ID low | vlan ID high }

import	Assigns or imports VLAN information to the current vsys. When a VLAN is assigned to a vsys, it cannot be shared with another vsys and remains assigned until the system admin unsets it. Note: The **set vlan import** command works only under a vsys.

port

get vlan port [all]
set vlan port *interface* { group *name_str* zone *zone* }
unset vlan port { *interface* group *name_str* }

port	Binds the VLAN group to the specified interface. You must also bind the group to a zone.

Example: The following command binds the VLAN group `secure_vlan` to interface `ethernet2/1` in the v1-trust zone:

set vlan port ethernet2/1 group secure_vlan zone v1-trust

retag

get vlan retag [name *string*]
set vlan retag { name *name_str number number* }
unset vlan retag { name *name_str* }

retag	Specifies that VLAN tags on traffic for the named VLAN be converted from ID *number* to ID *number*. Retagged traffic is diverted from the Layer 2 switch to the security device using the second tag ID. Retagging permits traffic that would be bound for a particular VLAN to be diverted without having to reconfigure policies or interfaces. Allowed VLAN IDs range from 2 to 4094.

Example: The following command creates a retagging pair called `secure_one` that retags traffic on interface `ethernet2/1` with VLAN ID 10 to ID 20 and retags VLAN traffic with ID 20 to ID 10 on interface `ethernet2/1`:

set vlan retag secure_one 10 20

set vlan port eth2/1 retag secure_one

zone

set vlan port *interface* { retag *name_str* }

zone	Assigns a Layer 2 zone to the VLAN group.

Example: The following command binds the VLAN group `secure_vlan` to interface `ethernet2/1` in the v1-trust zone:

set vlan port ethernet2/1 group secure_vlan zone v1-trust

vpn

Use the vpn commands to create or remove a Virtual Private Network (VPN) tunnel, or to display current VPN tunnel parameters.

A *tunnel* is a way to secure VPN communication across a WAN. The tunnel consists of a pair of unidirectional security associations (SAs), one at each end of the tunnel, that specify the security parameter index (SPI), destination IP address, and security protocol (Authentication Header or Encapsulating Security Payload) used to exchange packets through the tunnel.

security devices support two keying methods for establishing VPN tunnels, AutoKey IKE and Manual Key. AutoKey IKE (Internet Key Exchange) is a standard protocol that automatically establishes and maintains encryption keys between the participants. Manual Key VPNs use predefined keys that remain unchanged until the participants change them explicitly.

Syntax

get

```
get vpn [ name_str [ detail ] | auto | manual | proxy-id | ]
```

set (AutoKey IKE)

```
set vpn tunn_str gateway { ip_addr | name_str }
    [ replay | no-replay ]
      [ transport | tunnel ]
        [ idletime number ]
          {
          proposal [ name_str1 [ name_str2 [ name_str3 [ name_str4 ] ] ] ] |
          sec-level { basic | compatible | standard }
          }
```

set (Manual Key)

```
set vpn tunn_str manual spi_num1 spi_num2 gateway ip_addr
    [ outgoing-interface interface ]
      {
      ah { md5 | sha-1 }
        { key key_str | password pswd_str } |
      esp
        {
        aes128 | aes192 | aes256 | des | 3des
          { key key_str | password pswd_str } |
        null
        }
          [ auth { md5 | sha-1 }
            { key key_str | password pswd_str }
          ]
      }
```

set (Tunnel)

```
set vpn tunn_str
    {
    bind { interface interface | zone name_str } |
    df-bit { clear | copy | set } |
    failover-weight number
    monitor [ source-interface interface [ destination-ip ip_addr ] ]
      [ optimized } [ rekey ] |
    proxy-id local-ip ip_addr/mask remote-ip ip_addr/mask svc_name
    }
```

Keywords and Variables

Variables

get vpn *tunn_str* [...]

name_str	Defines a name for the VPN.

Example: The following command displays a VPN named “branch”:

get vpn branch

ah

set vpn *tunn_str* manual *spi_num1 spi_num2* gateway *ip_addr* [...] ah { ... }

ah	Specifies Authentication Header (AH) protocol to authenticate IP packet content. ■ md5 Specifies the Message Digest 5 (MD5) hashing algorithm. (128-bit) ■ sha-1Specifies the Secure Hash Algorithm (version) 1 (SHA-1) hashing algorithm. (160-bit) The key *key_str* value defines a 16-byte (MD5) or 20-byte (SHA-1) hexadecimal key, which the security device uses to produce a 96-bit message digest (or hash) from the message. password *pswd_str* Specifies a password the security device uses to generate an encryption or authentication key automatically.

Example: The following command creates a Manual Key VPN tunnel named “Mkt_vpn”.

- Sets the local and remote SPI values as 2002 and 3003
- Defines the remote gateway address 2.2.2.2
- Specifies Authentication Header (AH) protocol for IP packet authentication using the SHA-1 algorithm, the key for which is generated from the password “swordfish”

```
set vpn Mkt_vpn manual 2002 3003 gateway 2.2.2.2 ah sha-1 password swordfish
```

auto

get vpn auto

auto	Displays all AutoKey IKE VPNs.

Example: The following command displays all AutoKey IKE VPNs:

```
get vpn auto
```

bind

set vpn *tunn_str* bind { interface *interface* | zone *name_str* }
unset vpn *vpn_name* bind { interface | zone }

bind	Binds VPN tunnel to a tunnel interface or a security zone. ■ interface *interface* specifies the tunnel interface to use for VPN binding. ■ zone *name_str* specifies the tunnel zone to use for VPN binding.

Example: The following command binds the VPN tunnel named vpn1 to the tunnel.1 interface:

```
set vpn vpn1 bind interface tunnel.1
```

Example: The following command binds the VPN tunnel named vpn2 to the Untrust-Tun tunnel zone:

set vpn vpn2 bind zone untrust-tun

df-bit

set vpn *tunn_str* df-bit { clear **|** copy **|** set }

df-bit

Determines how the security device handles the Don't Fragment (DF) bit in the outer header.

- clear Clears (disables) DF bit from the outer header. This is the default value.
- copy Copies the DF bit to the outer header.
- set Sets (enables) the DF bit in the outer header.

esp

set vpn *tunn_str* manual *spi_num1 spi_num2* gateway *ip_addr* esp { … }

esp

Specifies the use of the Encapsulating Security Payload (ESP) protocol, which the security device uses to encrypt and authenticate IP packets.

- aes128 Specifies Advanced Encryption Standard (AES). The key *key_str* value defines a 128-bit hexadecimal key.
- aes192 Specifies Advanced Encryption Standard (AES). The key *key_str* value defines a 192-bit hexadecimal key.
- aes256 Specifies Advanced Encryption Standard (AES). The key *key_str* value defines a 256-bit hexadecimal key.
- des Specifies Data Encryption Standard (DES). The key *key_str* value defines a 64-bit hexadecimal key (truncated to 56 bits).
- 3des Specifies Triple Data Encryption Standard (3DES). The key *key_str* value defines a 192-bit hexadecimal key (truncated to 168 bits).
- null Specifies no encryption. (When you specify this option, you must specify an authentication algorithm (MD5 or SHA-1) using the auth option.)

auth Specifies the use of an authentication (hashing) method. The available choices are MD5 or SHA-1. (Some security devices do not support SHA-1.) The key *key_str* value defines a 16-byte (MD5) or 20-byte (SHA-1) hexadecimal key, which the security device uses to produce a 96-bit message digest (or hash) from the message.

password *pswd_str* Specifies a password the security device uses to generate an encryption or authentication key automatically.

Example: The following command creates a Manual Key VPN tunnel named “Mkt_vpn”.

- Specifies local and remote SPI values 2002 and 3003
- Specifies the IP address of the remote gateway 2.2.2.2
- Specifies ESP with 3DES encryption and SHA-1 authentication
- Generates the encryption and authentication keys from the passwords “swordfish” and “avalanche”

```
set vpn Mkt_vpn manual 2002 3003 gateway 2.2.2.2 esp 3des password swordfish
    auth sha-1 password avalanche
```

failover-weight

```
set vpn name_str failover-weight number }
```

failover-weight

Assigns a weight to a VPN tunnel. When the accumulated weight of failed or “down” VPN tunnels bound to the primary Untrust zone interface reaches or exceeds 100 percent, ScreenOS fails over to the backup Untrust zone interface.

Note: This option is available only on devices that support the DIAL-backup feature.

Example: The following command assigns a failover weight of 50 percent to the VPN to_remote1:

```
set vpn to_remote1 failover-weight 50
```

gateway

```
set vpn tunn_str gateway ip_addr [ … ] { … }
set vpn tunn_str gateway name_str [ … ] { … }
get vpn gateway [ detail ]
```

gateway

Specifies the autokey IKE gateway (*ip_addr* or *name_str*) to use.

- **idletime** *number* The length of time in minutes that a connection can remain inactive before the security device terminates it.
- **replay | no-replay** Enables or disables replay protection. The default setting is no-replay.
- **transport | tunnel** Defines the IPSec mode. In tunnel mode, the active IP packet is encapsulated. In transport mode, no encapsulation occurs. Tunnel mode is appropriate when both of end points in an exchange lie beyond gateway devices. Transport mode is appropriate when either end point is a gateway.
- **proposal** *name_str* Defines up to four Phase 2 proposals. A Phase 2 proposal determines how a security device sends VPN session traffic.
- **sec_level** Specifies a predefined set of proposals.

Example: In the following example you define an IKE gateway for a remote site in London. The gateway has the following elements:

- The remote gateway is named London_Office, with IP address 2.2.2.2.
- The outgoing interface is ethernet3.
- The Phase 1 proposal consists of the following components:
  - DSA certificate for data source authentication
  - Diffie-Hellman group 2 to protect the exchange of keying information
  - AES-128 encryption algorithm
  - MD-5 authentication algorithm

You then reference that gateway in a VPN tunnel that has the following elements:

- The tunnel is named London_Tunnel.
- The Phase 2 proposal consists of the following components:
  - Diffie-Hellman group 2 to protect the keying information during Phase 2 key exchanges
  - Encapsulating Security Payload (ESP) to provide both confidentiality through encryption and encapsulation of the original IP packet and integrity through authentication
  - AES-128 encryption algorithm
  - MD-5 authentication algorithm

```
set ike gateway London_Office ip 2.2.2.2 outgoing-interface ethernet3 proposal
    dsa-g2-aes128-md5
set vpn London_Tunnel gateway London_Office proposal g2-esp-aes128-sha
```

Example: In the following example you define IKE gateways between subnets (one at a San Francisco site and the other at a London site), then build a two-way VPN tunnel between them. Both gateway interfaces are in IPv6 mode.

1. The following sequence of commands, executed on the security device at the San Francisco site, define an IKE gateway to the IPv6 gateway at the London site, use the gateway to define a VPN tunnel, then create a policy that permits local hosts to request HTTP service through the tunnel.
   - The remote gateway is named London_Office, with IPv6 address 2345::200:ff:fe00:6.
   - The outgoing interface is ethernet3, with local IPv6 address 2e08::2200:e6ff:fe57.
   - The local host subnet IPv6 prefix is address-book entry SF_Local (21e4::e443:a1ff:feb6/64).

- The remote host subnet IPv6 prefix is address-book entry London_Remote (3a55::130:6:aff:fe3a/64).
- The VPN tunnel, which uses the London_Office gateway, is named London_Tunnel.

```
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode host
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 2e08::2200:e6ff:fe57/64
set ike gateway London_Office ip 2345::200:ff:fe00:6 outgoing-interface
    ethernet3 local-address 2e08::2200:e6ff:fe57 proposal dsa-g2-aes128-md5
set vpn London_Tunnel gateway London_Office proposal g2-esp-aes128-sha
set address trust SF_Local 21e4::e443:a1ff:feb6/64
set address untrust London_Remote 3a55::130:6:aff:fe3a/64
set policy from trust to untrust SF_Local London_Remote http tunnel vpn
    London_Tunnel
```

2. The following command, executed on the security device at the London site, defines an IKE tunnel to the IPv6 gateway at the San Francisco site.

- The remote gateway is named SF_Office, with IPv6 address 2e08::2200:e6ff:fe57.
- The outgoing interface is ethernet3, with local IPv6 address 2345::200:ff:fe00:6.
- The remote host subnet IPv6 prefix is address-book entry SF_Remote (21e4::e443:a1ff:feb6/64).
- The local host subnet IPv6 prefix is address-book entry London_Local (3a55::130:6:aff:fe3a/64).
- The VPN tunnel, which uses the SF_Office gateway, is named SF_Tunnel.

```
set interface ethernet3 zone untrust
set interface ethernet3 ipv6 mode host
set interface ethernet3 ipv6 enable
set interface ethernet3 ipv6 ip 2345::200:ff:fe00:6/64
set ike gateway SF_Office ip 2e08::2200:e6ff:fe57 outgoing-interface
    ethernet3 local-address 2345::200:ff:fe00:6 proposal dsa-g2-aes128-md5
set vpn SF_Tunnel gateway SF_Office proposal g2-esp-aes128-sha
set address trust London_Local 3a55::130:6:aff:fe3a/64
set address untrust SF_Remote 21e4::e443:a1ff:feb6/64
set policy from untrust to trust London_Remote London_Local http tunnel vpn
    SF_Tunnel
```

manual

get vpn *tunn_str* [detail] manual
set vpn *tunn_str* manual *spi_num1 spi_num2* gateway *ip_addr* […] { … }

manual	Specifies a Manual Key VPN. When the security device is in Manual mode, you can encrypt and authenticate by HEX key or password. *spi_num1* and *spi_num2* are 32-bit *local* and *remote* security parameters index (SPI) numbers. Each SPI number uniquely distinguishes a particular tunnel from any other active tunnel. Each must be a hexadecimal value between 3000 and 2fffffff. The local SPI corresponds to the remote SPI at the other end of the tunnel, and vice-versa.

monitor

set vpn *tunn_str* monitor […] [destination-ip *ip_addr*] […]
unset vpn *tunn_str* monitor

monitor	Directs the security device to send VPN monitor messages to a NetScreen-Remote client or a non-Juniper Networks peer device. The source-interface *interface* option specifies the interface through which the security device sends the monitor messages. ■ destination-ip specifies the destination IP address for the VPN monitoring feature to ping. ■ optimized performs optimization for scalability. ■ rekey triggers rekey of an autokey VPN is a tunnel is down.

Example: The following command uses ethernet3 as the source interface and 10.1.1.5 as the destination IP address for VPN monitoring through a VPN tunnel named tun1:

set vpn tun1 monitor source-interface ethernet3 destination-ip 10.1.1.5

outgoing-interface

set vpn *tunn_str* manual *spi_num1 spi_num2* gateway *ip_addr* […]
    outgoing-interface *interface* { … }

outgoing-interface	Defines the interface through which the security device sends traffic for this Manual Key VPN. For more information, see “Interfaces” on page 319.

proxy-id

get vpn proxy-id
set vpn *tunn_str* proxy-id local-ip *ip_addr* / *mask* remote-ip *ip_addr* / *mask svc_name*
unset vpn vpn_name proxy-id

proxy-id	Specifies the three-part tuple consisting of local IP address–remote IP address–service. ■ local-ip *ip_addr/mask* The local subnet, expressed as an IPv4 address and mask or an IPv6 address and prefix length (*ip_addr/pref_len*). ■ remote-ip *ip_addr/mask* The remote IP address that sends and receives traffic through the tunnel. ■ *svc_name* The name of the service, such as FTP, TELNET, DNS or HTTP that passes through the tunnel. (Specifying any enables all services.)

Example 1: The following command creates a VPN proxy configuration for a VPN (Sales) with the HTTP service:

set vpn Sales proxy-id local-ip 172.16.1.1 / 24 remote-ip 192.168.2.2 / 24 HTTP

Example 2: The following command creates a VPN proxy configuration for a VPN (Support) with the FTP service:

set vpn Support proxy-id local-ip 2eee:45::9 / 64 remote-ip 2::10:3 / 64 FTP

rekey

set vpn corp monitor rekey

rekey	Keeps the SA active even if there is no other VPN traffic.

sec-level

set vpn *tunn_str* gateway { *name_str* | *ip_addr* } […] { ... } sec-level
{ basic | compatible | standard }

sec-level	Specifies which predefined security proposal to use for IKE. The basic proposal provides basic-level security settings. The compatible proposal provides the most widely used settings. We recommend using the standard proposal setting.

vpn-group

Use the **vpn-group** commands to define or remove VPN groups, or to display VPN groups.

A *VPN group* is a collection of defined VPN tunnels. A VPN group allows the security device to perform tunnel failover. Each tunnel in the group has an assigned weight. When the security device invokes a policy that uses a VPN group, the device constructs all tunnels in the group, and the tunnel with the greatest weight becomes active by default. The IKE heartbeat periodically checks to see if this tunnel is working. If it is not, the device uses the tunnel with the next highest weight.

Syntax

get

get vpn-group [id *id_num*]

set

set vpn-group id *id_num* [vpn *tunn_str* [weight *number*]]

Keywords and Variables

id

get vpn-group id *id_num*
set vpn-group id *id_num* [...]
unset vpn-group id *id_num* [...]

id	Specifies an identification number for the VPN group.

vpn

set vpn-group id *id_num* vpn *tunn_str* [...]
unset vpn-group id *id_num* vpn *tunn_str*

vpn	Specifies the name of a VPN to place in the VPN group.

weight

set vpn-group id *id_num* vpn *tunn_str* weight *number*
unset vpn-group id *id_num* vpn *tunn_str* weight *number*

weight	Specifies a weight (priority) for the VPN relative to other VPNs in the group. The higher the number, the higher priority.

Example: With the following commands, you create two VPN tunnels (vpn1 and vpn2). You place them in a VPN group with ID 1001, which you then reference in a policy permitting traffic from addr1 in the Trust zone to addr2 in the Untrust zone beyond the remote gateway. You assign vpn1 a greater weight, giving it priority. If traffic cannot pass through vpn1, the security device redirects it through vpn2:

```
set ike gateway gw1 ip 1.1.1.1 preshare bi273T1L proposal pre-g2-3des-md5
set ike gateway gw2 ip 2.2.2.2 preshare r3ix6403 proposal pre-g2-aes128-md5
set vpn vpn1 gateway gw1 replay proposal g2-esp-3des-sha
set vpn vpn2 gateway gw2 replay proposal g2-esp-3des-sha
set vpn-group id 1001 vpn vpn1 weight 1
set vpn-group id 1001 vpn vpn2 weight 2
set policy from trust to untrust addr1 addr2 HTTP tunnel vpn-group 1001
```

vpnmonitor

Use the **vpnmonitor** commands to set the monitor frequency and threshold.

ScreenOS provides the ability to determine the status and condition of active VPNs through the use of ICMP pings, and report the conditions by using SNMP VPN monitoring objects and traps.

To enable your SNMP manager application to recognize the VPN monitoring MIBs, you must import the NetScreen-specific MIB extension files into the application. The MIB extension files are on the Juniper Networks documentation CD that shipped with the security device.

Syntax

get

```
get vpnmonitor
```

set

```
set vpnmonitor
    {
    interval number |
    threshold number
    }
```

Keywords and Variables

interval

```
set vpnmonitor interval number
unset vpnmonitor interval
```

interval	Specifies the monitor frequency interval (in seconds).

threshold

set vpnmonitor threshold *number*
unset vpnmonitor threshold

threshold	Specifies the monitor threshold, the number of consecutive times the device can send vpnmonitor requests without getting a response before the device changes the VPN Link-Status to down.

vrouter

Use the **vrouter** commands to configure a virtual router on the security device.

Executing the **set vrouter** *name_str* command without specifying further options places the CLI in the routing context. For example, the following command places the CLI in the **trust-vr** routing context:

set vrouter trust-vr

Once you initiate the routing context, all subsequent command executions apply to the specified local virtual router (**trust-vr** in this example). You can then initiate the **bgp**, **ospf**, **rip** or **ripng** protocol context.

- To enter the **bgp** context, execute the **set protocol bgp** command.

  device(trust-vr)-> **set protocol bgp**

- To enter the **ospf** context, execute the **set protocol ospf** command.

  device(trust-vr)-> **set protocol ospf**

- To enter the **rip** context, execute the **set protocol rip** command.

  device(trust-vr)-> **set protocol rip**

- To enter the **ripng** context, execute the **set protocol ripng** command.

  device(trust-vr)-> **set protocol ripng**

In the **bgp**, **ospf**, **rip** or **ripng** protocol context, all command executions apply to the specified protocol.

For more information about IPv4 **vrouter protocol** options, see the **bgp**, **ospf**, and **rip** command descriptions in the *ScreenOS CLI Reference Guide: IPv4 Command Descriptions*.

Syntax

exec

```
exec vrouter name_str protocol bgp neighbor ip_addr
    {
    connect |
    disconnect |
    tcp-connect
    }
```

get

```
get vrouter name_str
    [
    access-list |
    config |
    default-vrouter |
    interface |
    preference |
    protocol { bgp | ospf | rip | ripng } |
    route [ v4 | v6 ]
      [
      id id_num |
      ip ip_addr |
      prefix ip_addr/mask |
      protocol { bgp | connected | discovered | imported | ospf | rip | ripng | static }
      source [ id id_num | ip ip_addr | prefix ip_addr/mask ] |
      summary
      ] |
    route-map [ name_str ]
      [
      config |
      number [ config | match | set ]
      ] |
    router-id |
    rule |
    statistics |
    zone
    ]
```

set

```
set vrouter { name name_str | name_str }
    [
    access-list id_num [ ipv6 ]
        { permit | deny } { ip ip_addr/mask | default-route } number |
    add-default-route vrouter untrust-vr |
    adv-inact-interface
    auto-route-export |
    default-vrouter |
    enable-source-routing
    export-to | import-from
        vrouter name_str route-map name_str protocol
            { bgp | connected | discovered | imported | ospf | rip | ripng | static }
    ignore-subnet-conflict |
    max-routes number |
    nsrp-config-sync |
    preference
        {
        auto-exported number |
        connected number |
        ebgp number |
        ibgp number |
        imported number |
        ospf number |
        ospf-e2 number |
        rip number
        static number
        } |
    protocol { bgp | ospf | pim | rip | ripng } |
    route [ source ] ip_addr/mask
        {
        interface interface
            [ gateway ip_addr ] [ metric number ] [ tag id_num ] |
        vrouter name_str
        } |
    route-map
        {
        name name_str { permit | deny } number |
        name_str number }
            [
            as-path id_num |
            community id_num |
            local-pref number |
            match
                {
                as-path id_num |
                community id_num |
                interface interface |
                ip id_num |
                metric number |
                next-hop id_num |
                route-type
                    { internal-ospf | type1-external-ospf | type2-external-ospf } |
                tag { number | ip_addr }
                }|
            metric number |
          metric-type { type-1 | type-2 } |
          next-hop ip_addr |
          offset-metric number |
          tag { number | ip_addr } |
          weight number
          ]
     router-id { id_num | ip_addr } |
     sharable
     snmp trap private
     ]
```

Keywords and Variables

Variables

set vrouter *name_str*

name_str The name of the virtual router. The name can be a predefined virtual router, such as trust-vr or untrust-vr, or it can be a user-defined virtual router created with the name keyword. (Creating custom virtual routers is only supported on certain security devices and requires a vsys software key.).

Example: The following commands activate the trust-vr virtual router context, activate the IPv6 RIPng routing context, and execute the context-dependent command get config.

```
set vrouter trust-vr
device(trust-vr)-> set protocol ripng
device(trust-vr/ripng)-> get config
```

access-list

```
get vrouter name_str access-list
set vrouter name_str access-list id_num
     { permit | deny } { ip ip_addr/mask | default-route } number }
set vrouter vrouter access-list ipv6 id_num [ { permit | deny } { ... } ]
unset vrouter name_str access-list id_num [ ip_addr/mask | default-route ] number
```

access-list Creates or removes an access list, or entries in the access list. Each entry permits (or denies) routes according to IP address and mask, or default route. The *id_num* value identifies the access list. The optional ipv6 switch indicates that the protocol is IPv6. The number identifies the sequence number for this entry in the access list.

- permit Directs the virtual router to permit the route.
- deny Directs the virtual router to deny the route.
- default-route Enters the default route for the virtual router into the access list.

Note: An access list can contain only IPv4 or IPv6 addresses, and cannot combine them. To create an IPv4 access list, use access-list. To create an IPv6 access list, use access-list ipv6.

add-default-route

set vrouter *name_str* add-default-route vrouter *name_str*
unset vrouter *name_str* add-default-route

add-default-route	Adds a default route with the next hop as another virtual router. (This command is available only in the default virtual router of the current vsys, and only if this virtual router is not untrust-vr.)

adv-inact-interface

set vrouter *name_str* adv-inact-interface
unset vrouter *name_str* adv-inact-interface

adv-inact-interface	Directs the virtual router to consider active routes on inactive interfaces for redistribution or export. By default, only active routes defined on active interfaces can be redistributed to other protocols or exported to other virtual routers.

auto-route-export

set vrouter *name_str* auto-route-export
unset vrouter *name_str* auto-route-export

auto-route-export	Directs the virtual router to export public interface routes to the untrust-vr vrouter. An interface is public if it is in Route mode, and private if it is in NAT mode. For information about Route and NAT modes, refer to the *Concepts and Examples ScreenOS Reference Guide*. Note: The auto-route-export switch does not take effect if the specified vrouter (*name_str*) has export or import rules to the untrust-vr virtual router.

config

get vrouter *name_str* config

config	Displays configuration information about the virtual router.

default-vrouter

get vrouter *name_str* default-vrouter
set vrouter *name_str* default-vrouter

default-vrouter	Sets the specified virtual router as the default router for the vsys.

enable-source-routing

get vrouter *name_str* enable-source-routing
set vrouter *name_str* enable-source-routing

enable-source-routing

Directs the virtual router to perform routing table lookups based on source IP address.

export-to | import-from

set vrouter *name_str* { export-to | import-from } vrouter *name_str* { … }
unset vrouter *name_str* { export-to | import-from } vrouter *name_str* { … }

export-to | import-from

Directs the virtual router to import routes from another virtual router (source), or to export routes to another virtual router (destination).

- vrouter *name_str* identifies the source or destination virtual router.
- route-map *name_str* identifies the route map that filters the imported or exported routes.
- protocol Specifies the protocol for the imported or exported routes.
- bgp Directs the virtual router to import or export Border Gateway Protocol (BGP) routes.
- connected Directs the virtual router to import or export connected routes.
- imported Directs the virtual router to import or export routes that were redistributed into the virtual router from another virtual router.
- ospf Directs the virtual router to import or export Open Shortest Path First (OSPF) routes.
  - rip Directs the virtual router to import or export Routing Information Protocol (RIP) routes.
  - ripng Directs the virtual router to import or export Routing Information Protocol next-generation (RIPng) routes. RIPng is for IPv6 protocol.
  - static Directs the virtual router to import or export static routes.
- default-route Directs the virtual router to export or import the default route.

ignore-subnet-conflict

set vrouter *name_str* ignore-subnet-conflict
unset vrouter *name_str* ignore-subnet-conflict

ignore-subnet-conflict	Directs the virtual router to ignore overlapping subnet addresses for interfaces in the virtual router. By default, you cannot configure overlapping subnet IP addresses on interfaces in the same virtual router.

interface

get vrouter *name_str* interface

interface	Displays the interfaces in the virtual router.

max-routes

set vrouter *name_str* max-routes *number*
unset vrouter *name_str* max-routes

max-routes	Specifies the maximum number of routing entries allowed for this virtual router. By default, the maximum number of entries allowed for a virtual router depends upon the security device and the number of virtual routers configured on the device.

name

set vrouter name *name_str*

name	Specifies the name of a user-defined virtual router. Creating custom virtual routers is only supported on certain security devices and requires a vsys software key.

nsrp-config-sync

set vrouter *name_str* nsrp-config-sync
unset vrouter *name_str* nsrp-config-sync

nsrp-config-sync	Synchronizes the specified virtual router (*name_str*) with the same virtual router on an NSRP peer. This switch is enabled by default.

preference

get vrouter *name_str* preference
set vrouter *name_str* preference
unset vrouter *name_str* preference

preference

Specifies route preference level based upon protocol. The lower the value, the more preference given to the route. You can specify a value between 1-255.

- **auto-exported** Specifies preference levels for routes (defined on public interfaces) that the virtual router automatically exports to the untrust-vr virtual router. The default is 30.
- **connected** Specifies preference level for connected routes. The default is 0.
- **ebgp** Specifies preference level for External Border Gateway Protocol (EBGP) routes. The default is 120.
- **ibgp** Specifies preference level for Internal Border Gateway Protocol (IBGP) routes. The default is 40.
- **imported** Specifies preference level for preexisting routes exported to another protocol and passed on to other routers. The default is 140.
- **ospf** Specifies preference level for Open Shortest Path First (OSPF) routes. The default is 60.
- **ospf-e2** Specifies preference level for OSPF External Type 2 routes. The default is 200.
- **rip** Specifies preference level for Routing Information Protocol (RIP) routes. The default is 100.
- **ripng** Specifies preference level for Routing Information Protocol next-generation (RIPng) routes. (RIPng is for IPv6 protocol.) The default is 100.
- **static** Specifies preference level for static routes. The default is 20.

protocol

exec vrouter *name_str* protocol { ... }
get vrouter *name_str* protocol { bgp **|** ospf **|** rip }
set vrouter *name_str* protocol { bgp **|** ospf **|** rip }
unset vrouter *name_str* protocol { bgp **|** ospf **|** rip }

protocol

Places the security device in the BGP context or the OSPF context. (For information about these contexts, see the **bgp**, **ospf**, or **rip** command descriptions in this manual.)

The **exec vrouter** *name_str* **protocol bgp neighbor** *ip_addr* command has the following options:

- **connect** Establishes a BGP connection to the specified neighbor.
- **disconnect** Terminates a BGP connection to the specified neighbor.
- **tcp-connect** Tests the TCP connection to the neighbor.

route

get vrouter *name_str* route […]
set vrouter *name_str* route [source] *ip_addr/mask* { interface *interface* [gateway *ip_addr* [metric *number*] [tag *id_num*]] **|** vrouter *name_str* }
unset vrouter *name_str* route [source] *ip_addr/mask* vrouter *name_str* interface *interface* [gateway *ip_addr*]]

route

Configures routes for the routing table for the virtual router.

- *ip_addr/ { mask | pref_len }*Specifies the IPv4 address and subnet mask or IPv6 address and prefix length that appears as an entry in the routing table.
- gateway *ip_addr* Specifies the IPv4 address and subnet mask or IPv6 address gateway for the next hop. The metric value specifies the cost of the route.
- id *id_num* Displays information for the route that matches the ID number. The ID number is a system-assigned number that you can see when you enter the get vrouter *name_str* route command with no options.
- interface *interface* Specifies the routed interface.
- ip *ip_addr* Displays the route for the specified IPv4 or IPv6 address.
- prefix *ip_addr/mask* Displays the routes within the specified IPv4 address and subnet mask or IPv6 address and prefix length.
- protocol Displays BGP, connected, imported, OSPF, RIP, or static routes.
- source Specifies that the route is a source-based route. When displaying a source-based route, you can optionally specify:
  - id *id_num*
  - ip *ip_addr*
  - prefix *ip_addr/mask*
- summary Displays a summary of the routes.
- vrouter *name_str* Specifies a virtual router as the next hop.

route-map

get vrouter *name_str* route-map […]
set vrouter *name_str* { … } vrouter *name_str* route-map { name *name_str* **|** *name_str* } […]
set vrouter *vrouter* { … } vrouter *vrouter* route-map ipv6 name *name_str* […]
unset vrouter *name_str* { … } vrouter *name_str* route-map *name_str* […]

route-map

Configures a route map for the virtual router.

With the name keyword, the route-map option creates a new route map (*name_str*). Otherwise, *name_str* configures an existing route map. Each entry in the route map must have a sequence number (*number*) that identifies the order in which the route map entries are compared against an incoming or outgoing route. The permit and deny switches determine if the entry allows redistribution of routes to another virtual router or another protocol.

The ipv6 switch identifies the route as an IPv6 route.

The **match** keyword directs the virtual router to match routes to specified parameters. You can match the following parameters:

- **as-path** *id_num* Specifies an AS path access list that defines the BGP AS path attribute to be matched.
- **community** *id_num* Specifies a BGP community list (*id_num*) that defines the community attribute to be matched.
- **interface** *interface* Specifies an interface on the security device.
- **ip** *id_num* Specifies an access list that defines the IP addresses of routes to be matched.
- **metric** *number* The cost of the route. Enter a number between 1-65535.
- **next-hop** *id_num* Specifies an access list that defines the next-hop for routes to be matched
- **route-type** Specifies which kind of OSPF route matches the route map entry.
  - **internal-ospf** Matches only OSPF internal routes.
  - **type1-external-ospf** Matches only external OSPF Type-1 routes.
  - **type2-external-ospf** Matches only external OSPF Type-2 routes.
- **tag** { *number* | *ip_addr* } Matches either a route tag or an IP address.

Other keywords allow you to optionally set values for parameters on matching routes. You can set the following parameters:

- **as-path** *id_num* Specifies the AS path access list values that are prepended to the path list of the matching route.
- **community** *id_num* Specifies the community list values that are set in the community attribute for the matching route.
- **local-pref** *number* Specifies the path preference for the matching route.
- **metric** *number* Specifies the metric for the matching route. Enter a number between 1-65535.
- **metric-type** Specifies OSPF metric type that is set for the matching route.
  - **type-1** Specifies OSPF Type-1 route.
  - **type-2** Specifies OSPF Type-2 route.
- **next-hop** *ip_addr* Specifies the next hop IP address for the matching route.
- **offset-metric** *number* Specifies the RIP offset to increase the metric for the matching route.
- **tag** { *number* | *ip_addr* } Specifies a tag or IP address for the matching route.
- **weight** *number* Sets the weight of the matching route for BGP.

While configuring a route map, you can use the get **config**, **get match**, and **get set** commands to display route map configuration commands, or match or set conditions.

router-id

```
get vrouter name_str router-id
set vrouter name_str router-id { id_num | ip_addr }
unset vrouter name_str router-id
```

router-id Specifies the router identification that the virtual router uses to communicate with other routing devices. You can enter the router identification in either a dotted decimal notation (like an IP address) or a decimal number (this is converted to 0.0.0.*number*). If you do not specify a router identification, the device uses the highest IP address of the any interface in the virtual router as the router identification.

rule

```
get vrouter name_str rule
```

rule Displays import and export rules for the virtual router.

sharable

```
set vrouter name_str sharable
unset vrouter name_str sharable
```

sharable Makes the root-level virtual router accessible from any virtual system (vsys) on the device.

snmp

```
set vrouter name_str snmp trap private
unset vrouter name_str snmp trap private
```

snmp Makes SNMP traps private for the dynamic routing MIBs under the virtual router. Private traps include the virtual router identification.This option is available only for the default root-level virtual router. (This is usually the trust-vr virtual router, although you can change the default virtual router at the root level.)

statistics

```
get vrouter name_str statistics
```

statistics Displays statistics for the virtual router.

zone

```
get vrouter name_str zone
```

zone Displays the zones bound to the virtual router.

vsys

Use the **vsys** commands to create and configure virtual systems from the root level of a security device.

Virtual systems allow you to logically partition a single Juniper Networks security system to provide multi-tenant services. Each virtual system (vsys) is a unique security domain and can have its own administrators, called *virtual system administrators* or *vsys admins*. Such administrators can individualize their security domain by setting their own address books, virtual routers, user lists, custom services, VPNs, and policies. (Only a root-level administrator can set firewall security options, create virtual system administrators, and define interfaces and subinterfaces.)

When you execute the **set vsys** command, the command prompt changes to indicate that you are now operating within a virtual system. Use the **unset vsys** command to remove a specific virtual system and all its settings.

Syntax

get

```
get vsys [ name_str | cpu-limit | override | session-limit ]
```

set

```
set vsys name_str
    [
    vrouter
      [
      name [ name_str ] [ id id_num ] [ vsd number ] |
      share [ name_str ] [ vsd number ] |
      vsd number
      ] |
    vsd number |
    vsys-profile name_str
    ]
```

Keywords and Variables

Variables

```
get vsys [ name_str ]
set vsys name_str
unset vsys name_str
```

name_str	Defines the name of a virtual system and automatically places the root level admin within the virtual system. Subsequent commands configure the newly created virtual system.

Example: The following command creates a virtual system named **vsys1** and switches the console to the new virtual system:

set vsys vsys1

cpu-limit

```
get vsys cpu-limit
```

cpu-limit	Displays the CPU limit feature parameters for all virtual systems.

override

```
get vsys override
```

override	Displays the override values for all virtual systems.

session

```
get vsys session-limit
```

session-limit	Displays the maximum and reserved session values, sessions used and available, and alarm information. If maximum or reserved session values or alarm limit value has been overridden, the override value is shown.

vrouter

```
set vsys name_str vrouter [ name [ name_str ] [ id id_num ] [ vsd number ] ]
set vsys name_str vrouter [ share [ name_str ] [ vsd number ] ]
```

vrouter

Defines and configures the default virtual router for the vsys.

- name Specifies a name for the virtual router.
  - id *id_num* Assigns an identification number to the virtual router.
  - vsd *id_num* See “vsd” on page 299.
- share Specifies a shared root-level virtual router to use as a default router.
- vsd *id_num* See “vsd” on page 299.

Example 1: The following command creates a virtual system named Acme_Org, creates a virtual router named Acme_Router with ID 1025, and switches the console to the new virtual system:

```
set vsys Acme_Org vrouter name Acme_Router id 1025
```

Example 2: The following command creates a virtual system named Acme_Org, and specifies a default, root-level virtual router (trust-vr):

```
set vsys Acme_Org vrouter share trust-vr
```

vsd

set vsys *name_str* vrouter [vsd *number*]

vsd *number*	Assigns a Virtual Security Device (VSD) group number to the virtual router. A VSD group is a pair of physical security devices (a primary and a backup) that collectively comprise a single VSD. A VSD provides failover capability, allowing the backup device to take over if the primary device fails. For more information about VSD groups, refer to the *Concepts & Examples ScreenOS Reference Guide*.

Example: The following command creates a virtual system named Acme_Org, creates a virtual router named Acme_Router, creates a VSD ID 5, and switches the console to the new virtual system:

```
set vsys Acme_Org vrouter vsd 5
```

vsys-profile

set vsys *name_str* vsys-profile *name_str*

vsys-profile	Assigns an existing vsys profile to the vsys. The vsys profile must be previously defined.

vsys-profile

Use the **vsys-profile** commands to configure virtual system (vsys) profiles. Vsys profiles allow you to define resource allocation for individual virtual systems by setting a maximum value and reserved value for resources. The absolute maximum value for a resource depends on the security device, and the configured maximum value cannot exceed the device's absolute maximum value. The reserved value cannot be higher than the maximum value.

Use the **get vsys-profile** command to see a list of all vsys profiles and resource allocation information for each vsys profile.

Syntax

get

```
get vsys-profile [ name_str | global ]
```

set

```
set vsys-profile { name_str | name name_str }
    [ cpu-weight number ]
    [ dips
      {
      max number |
      reserve number
      }
    ]
    [ mips
      {
      max number |
   reserve number
   }
]
[ mpolicies
   {
   max number |
   reserve number
   }
]
[ policies
   {
   max number |
   reserve number
   }
]
[ sessions
   {
   alarm number
   max number |
   reserve number
   }
]
[ user-serv-grps
   {
   max number |
   reserve number
   }
]
[ user-servs
   {
   max number |
   reserve number
   }
]
[ user-zones
   {
   max number |
   reserve number
   }
]
[ zone-addr-grps
   {
   max number |
   reserve number
   }
]
[ zone-addrs
   {
   max number |
   reserve number
   }
]
```

Keywords and Variables

Variables

get vsys-profile [*name_str*]
set vsys-profile *name_str* [...]
unset vsys-profile *name_str*

name_str	Name of an existing virtual system (vsys).

Example: The following command configures a session limit of 500 for the existing vprofile1 profile:

set vsys-profile vprofile1 sessions max 500

cpu-weight

set vsys-profile { *name_str* | name *name_str* } cpu-weight *weight*

cpu-weight — Specifies CPU weight, which is a dimensionless quantity used to calculate the CPU time quota for each vsys. The CPU weight for a vsys is used in combination with the CPU weight for all the other virtual systems in a security device when calculating the time quota.

CPU weight can be a value from 1 through 100. The default value is 50.

Example: The following command configures a CPU weight of 30 for the existing vprofile1 profile:

set vsys-profile vprofile1 cpu-weight 30

dips

set vsys-profile { *name_str* | name *name_str* } dips { max *number* | reserve *number* }

max	Maximum number of dynamic IP addresses (DIPs) per vsys.
reserve	Number of DIPs reserved per vsys.

Example: The following command configures a maximum value of 200 for the existing vprofile1 profile:

set vsys-profile vprofile1 dips max 200

global

get vsys-profile [global]

global	Displays summary of global usage for the whole device.

mips

set vsys-profile { *name_str* **|** name *name_str* } mips { max *number* **|** reserve *number* }

max	Maximum number of mapped IP addresses (MIPs) per vsys.
reserve	Number of MIPs reserved per vsys.

Example: The following command configures a reserved value of 500 MIPs for the existing vprofile1 profile:

set vsys-profile vprofile1 mips reserve 500

mpolicies

set vsys-profile { *name_str* **|** name *name_str* } mpolicies { max *number* **|** reserve *number* }

max	Maximum number of multicast policies per vsys.
reserve	Number of multicast policies reserved per vsys.

Example: The following command configures a maximum value of 300 for the existing vprofile1 profile:

set vsys-profile vprofile1 mpolicies max 300

name

set vsys-profile name *name_str*

name	Name of the vsys profile. The maximum name length is 31 alphanumeric characters, including hyphens (-) and underscores (_). Spaces and special characters are not permitted.

Example: The following command creates a vsys profile named vprofile1:

set vsys-profile name vprofile1

policies

set vsys-profile { *name_str* **|** name *name_str* } policies { max *number* **|** reserve *number* }

max	Maximum number of security policies per vsys.
reserve	Number of security policies reserved per vsys.

Example: The following command configures a reserved value of 10000 policies for the existing vprofile1 profile:

set vsys-profile vprofile1 policies max 10000

sessions

set vsys-profile { *name_str* | name *name_str* } sessions { alarm *number* | max *number* | reserve *number* }

alarm	Number of sessions reached before an alarm is triggered.
max	Maximum number of sessions per vsys.
reserve	Number of sessions reserved per vsys.

Example: The following command configures a session limit of 500 for the existing vprofile1 profile:

```
set vsys-profile vprofile1 sessions 500
```

user-serv-grps

set vsys-profile { *name_str* | name *name_str* } user-serv-grps { max *number* | reserve *number* }

max	Maximum number of user service groups per vsys.
reserve	Number of user service groups reserved per vsys.

Example: The following command configures a maximum value of 500 user service groups for the existing vprofile1 profile:

```
set vsys-profile vprofile1 user-serv-grps max 500
```

user-servs

set vsys-profile { *name_str* | name *name_str* } user-servs { max *number* | reserve *number* }

max	Maximum number of user services per vsys.
reserve	Number of user services reserved per vsys.

Example: The following command configures a maximum value of 400 user services for the existing vprofile1 profile:

```
set vsys-profile vprofile1 user-servs max 400
```

user-zones

set vsys-profile { *name_str* | name *name_str* } user-zones { max *number* | reserve *number* }

max	Maximum number of zones per vsys.
reserve	Number of zones reserved per vsys.

Example: The following command configures a maximum value of 450 user service groups for the existing vprofile1 profile:

set vsys-profile vprofile1 user-zones max 450

zone-addr-grps

set vsys-profile { *name_str* **|** name *name_str* } zone-addr-grps { max *number* **|** reserve *number* }

max	Maximum number of zone address groups per vsys.
reserve	Number of zone address groups reserved per zone per vsys.

Example: The following command configures a reserved value of 1000 zone address groups for the existing vprofile1 profile:

set vsys-profile vprofile1 zone-addr-grps reserve 1000

zone-addrs

set vsys-profile { *name_str* **|** name *name_str* } zone-addrs { max *number* **|** reserve *number* }

max	Maximum number of zone addresses per vsys.
reserve	Number of zone addresses reserved per zone per vsys.

Example: The following command configures a maximum value of 15000 user zone addresses for the existing vprofile1 profile:

set vsys-profile vprofile1 zone-addrs max 15000

webauth

Use the **webauth** commands to configure the security device to perform WebAuth authentication.

WebAuth is an authentication method that requires the user to initiate an HTTP session and input authentication information, before the user can send traffic to the destination node.

You must specify authentication in policy definitions (see the auth keyword description in "policy" on page 195).

Syntax

get

```
get webauth [ banner ]
```

set

```
set webauth { banner success string | server name_str }
```

Keywords and Variables

banner success

```
get webauth banner
set webauth banner success string
unset webauth banner success
```

banner success	Specifies the banner (*string*) displayed in response to Webauth success.

Example: The following command changes the Webauth success banner to "Webauth service successful":

set webauth banner success "Webauth service successful"

server

```
set webauth server name_str
unset webauth banner server
```

server	Specifies the Webauth server name (*name_str*). (You can obtain all existing Webauth server names by executing the command get auth-server all.)

Example: The following command specifies a Webauth server named "Our_Webauth":

set webauth server Our_Webauth

Defaults

The default banner value is Webauth Success.

webtrends

Use the **webtrends** commands to configure the security device for WebTrends.

The WebTrends Firewall Suite allows you to customize syslog reports of critical, alert, and emergency events to display the information you want in a graphical format. You can create reports that focus on areas such as firewall attacks (emergency-level events) or on all events with the severity levels of critical, alert, and emergency.

Syntax

get

```
get webtrends
```

set

```
set webtrends
    {
    VPN |
    enable |
    host-name name_str |
    port port_num
    }
```

Keywords and Variables

vpn

```
set webtrends VPN
unset webtrends VPN
```

vpn	Enables WebTrends VPN encryption.

enable

```
set webtrends enable
unset webtrends enable
```

enable	Enables WebTrends.

host-name

```
set webtrends host-name name_str
unset webtrends host-name
```

host-name	Specifies the WebTrends hostname.

port

```
set webtrends port port_num
unset webtrends port
```

port *port_num*	Specifies the WebTrends host port.

xauth

Use the **xauth** commands to configure the security device to perform XAuth authentication.

An XAuth user or user group is one or more remote users who authenticate themselves when connecting to the security device via an AutoKey IKE VPN tunnel and optionally receive TCP/IP settings from the security device. Whereas IKE user authentication is actually the authentication of VPN gateways or clients, XAuth user authentication is the authentication of the users themselves. It requires each user to enter information unique to that user (the username and password).

Syntax

get

```
get xauth { active | default | lifetime }
```

set

```
set xauth
    {
    default
      {
      auth server name_str [ chap ] [ query-config ] |
      dns1 ip_addr | dns2 ip_addr |
      ippool name_str |
      wins1 ip_addr | wins2 ip_addr
      } |
    lifetime number
    }
```

Keywords and Variables

active

```
get xauth active
```

active	Displays all currently active XAuth login instances.

default

```
get xauth default
set xauth default { ... }
unset xauth default { ... }
```

default	Sets or displays default XAuth settings. ■ **auth server** Identifies the XAuth server by object name (*name_str*). ■ **chap** Directs the device to use Challenge Handshake Authentication Protocol (CHAP) while performing authentication with the XAuth client. ■ **query-config** Sets query client settings (such as IP address) from the external authentication server. ■ **dns1** Identifies the DNS primary server by IP address (*ip_addr*). ■ **dns2** Identifies the DNS secondary server by IP address (*ip_addr*). ■ **ippool** Identifies the IP pool (*name_str*). ■ **wins1** Identifies the WINS primary server by IP address (*ip_addr*). ■ **wins2** Identifies the WINS secondary server by IP address (*ip_addr*).

Example: The following command sets up the security device to use a XAuth server (Our_Auth):

set xauth default auth server Our_Auth

lifetime

```
get xauth lifetime
set xauth lifetime number
unset xauth lifetime number
```

lifetime *number*	Specifies the maximum length of time (in minutes) that the XAuth server holds resources (such as IP address) on behalf of the client.

Example: The following command specifies a maximum XAuth session length of 30 minutes:

set xauth lifetime 30

xlate

Use the **xlate** command to view translation transmission information. This command displays the following details:

- IP address
- Port number
- Translated IP address
- Translated port number
- Port count

- Dynamic IP (DIP) ID
- Count

Syntax

```
get xlate
```

Keywords and Variables

None.

zone

Use the **zone** commands to create, remove, or display a security zone, and to set SCREEN options.

A *security zone* is method for sectioning the network into segments to which you can apply various security options. You can configure multiple security zones for individual security devices, thus dividing the network into segments to which you can apply security options. There must be at least two security zones per device, basically to protect one area of the network from the other. On some platforms, you can define many security zones, bringing finer granularity to your network security design, without deploying multiple security appliances.

Each security zone has at least one interface bound to it. For a brief description of the interfaces, see “Interfaces” on page 319. For information about security zones, see “Zones” on page 321.

Syntax

get

```
get zone
    [
    id id_num |
    all |
    zone [ screen { all | attack | counter | info } ]
    ]
```

set

```
set zone
    {
    name zone [ { L2 id_num | tunnel zone } ] |
    zone
      {
      asymmetric vpn |
      block |
      screen
        {
        alarm-without-drop |
        block-frag |
        component-block [ activex | java | zip | exe ] |
        fin-no-ack |
    icmp-flood [ threshold number ] |
    icmp-fragment |
    icmp-large |
    ip-bad-option |
    ip-filter-src |
    ip-loose-src-route |
    ip-record-route |
    ip-security-opt |
    ip-spoofing [ drop-no-rpf-route ] |
    ip-stream-opt |
    ip-strict-src-route |
    ip-sweep [ threshold number ] |
    ip-timestamp-opt |
    land |
    limit-session
      [ source-ip-based number | destination-ip-based [ number ] ] |
    mal-url { string1 string2 number | code-red } |
    on tunnel |
    ping-death |
    port-scan [ threshold number ] |
    syn-ack-ack-proxy [ threshold number ] |
    syn-fin |
    syn-flood
      [
      alarm-threshold number |
      attack-threshold number |
      destination-threshold number |
      drop-unknown-mac |
      queue-size number |
      source-threshold number |
      timeout number
      ] |
    syn-frag |
    tcp-no-flag |
    tear-drop |
    udp-flood [ threshold number ] |
    unknown-protocol |
    winnuke
    }
  reassembly-for-alg |
  tcp-rst |
  vrouter name_str
  } |
```

Keywords and Variables

Variables

get zone *zone* [...]
set zone *zone* { ... }
unset zone *zone* { ... }

zone — The name of the zone. For more information, see “Zones” on page 321.

all

get zone all […]

all	Displays information about all existing zones.

asymmetric vpn

set zone asymmetric vpn

asymmetric vpn	When enabled, this option allows any incoming VPN traffic in a zone to match any applicable VPN session, regardless of the origin of the original VPN tunnel. For example, traffic coming from VPN A can match a session created by traffic for VPN B. This feature allows free routing of VPN traffic between two or more sites when there are multiple possible paths for VPN traffic. Note: It is not advisable to mix policy-based and route-based VPNs for asymmetric traffic.

block

set zone *zone* block
unset zone *zone* block

block	Imposes intra-zone traffic blocking.

name

set zone *name zone* { … }

name	Creates a new zone with name *zone*. ■ *L2 id_num* specifies that the zone is Layer 2 (for running the device in Transparent Mode). The ID number (*id_num*) identifies the VLAN to which the zone is bound. The name you specify (*zone*) must begin with “L2-”. ■ tunnel *zone* specifies that the new zone is a VPN tunnel zone, and identifies the tunnel-out zone (*zone*).

Example 1: The following command creates a new Layer 2 zone named L2-Sales, with VLAN ID number 1:

```
set zone name L2-Sales L2 1
```

Example 2: The following command creates a tunnel zone named Engineering and specifies untrust as the out zone:

```
set zone name Engineering tunnel untrust
```

reassembly-for-alg

set zone untrust reassembly-for-alg

reassembly-for-alg Reassembles all fragmented IP packets and TCP segments for HTTP and FTP traffic that arrives at any interface bound to the zone on which you enable this option. With this option enabled, the security device can better detect malicious URLs that an attacker has deliberately broken into packet or segment fragments. Packet and segment reassembly also improves application layer gateway (ALG) filtering by allowing the security device to examine the complete text within payloads.

screen

set zone *zone* screen { ... }
set zone *zone* screen { ... }

screen Enables or disables firewall services through the interface.

- **alarm-without-drop** Generates an alarm when detecting an attack, but does not block the attack. This option is useful if you allow the attack to enter a segment of your network that you have previously prepared to receive it—such as a honeynet, which is essentially a decoy network with extensive monitoring capabilities. It does not apply to traffic effected by the following features when the features have been enabled:
  - Session limit
  - SYN-ACK-ACK proxy protection
  - Malicious URL protection
- **block-frag** Enables IP packet fragmentation blocking.
- **component-block** Selectively blocks HTTP traffic containing any of the following components:
  - **activex** ActiveX controls
  - **java** Java applets
  - **exe** .EXE files
  - **zip** ZIP files

  An attacker can use any of these components to load an application (a Trojan Horse) on a protected host, then use the application to gain control of the host. If you enable the blocking of HTTP components without specifying which components, the security device blocks them all. Alternatively, you can configure the security device to block only specified components.

Note: If you enable ActiveX-blocking, the security device also blocks packets containing Java applets, .exe files, and .zip files because they might be contained within an ActiveX control.

- fin-no-ack Detects an illegal combination of flags, and rejects packets that have them.
- **icmp-flood** [**threshold** *number*] Detects and prevents Internet Control Message Protocol (ICMP) floods. An ICMP flood occurs when ICMP echo requests are broadcast with the purpose of flooding a system with so much data that it first slows down, and then times out and is disconnected. The threshold defines the number of ICMP packets per second allowed to ping the same destination address before the security device rejects further ICMP packets. The range is 1 to 1,000,000.
- **icmp-fragment** Detects and drops any ICMP frame with the More Fragments flag set, or with an offset indicated in the offset field.

- **icmp-large** Detects and drops any ICMP frame with an IP length greater the 1024.
- **ip-bad-option** Detects and drops any packet with an incorrectly formatted IP option in the IP packet header. The security device records the event in the SCREEN counters list for the ingress interface.
- **ip-filter-src** Detects and drops all packets with the Source Route Option enabled. The Source Route Option can allow an attacker to use a false IP address to access a network, and receive returned traffic addressed to the real IP address of the attacker's host device. The administrator can block all IP Source Routed frames having Strict Source Routing (or Loose Source Routing) enabled.
- **ip-loose-src-route** Detects packets where the IP option is 3 (Loose Source Routing) and records the event in the SCREEN counters list for the ingress interface. This option specifies a partial route list for a packet to take on its journey from source to destination. The packet must proceed in the order of addresses specified, but it is allowed to pass through other routers in between those specified.
- **ip-record-route** Detects packets where the IP option is 7 (Record Route) and records the event in the SCREEN counters list for the ingress interface.
- **ip-security-opt** Detects packets where the IP option is 2 (security) and records the event in the SCREEN counters list for the ingress interface.
- **ip-spoofing** Prevents spoofing attacks. Spoofing attacks occur when unauthorized agents attempt to bypass firewall security by imitating valid client IP addresses. Using the **ip-spoofing** option invalidates such false source IP address connections. Only security devices running in NAT or Route mode can use this option. The **drop-no-rpf-route** option instructs the security device to drop any packet that is not contained in the route table, for example, the device drops the packet if it does not contain a source route, or if the source IP address is reserved (non-routable, as with 127.0.0.1).
- **ip-stream-opt** Detects packets where the IP option is 8 (Stream ID) and records the event in the SCREEN counters list for the ingress interface.
- **ip-strict-src-route** Detects packets where the IP option is 9 (Strict Source Routing) and records the event in the SCREEN counters list for the ingress interface. This option specifies the complete route list for a packet to take on its journey from source to destination. The last address in the list replaces the address in the destination field.
- **ip-sweep threshold** *number* Detects and prevents an IP Sweep attack. An IP Sweep attack occurs when an attacker sends ICMP echo requests (pings) to multiple destination addresses. If a target host replies, it reveals the target's IP address to the attacker. Set the IP Sweep threshold to between 1 and 1,000,000 microseconds. Each time ICMP echo requests occur with greater frequency than this limit, the security device drops further echo requests from the remote source address.
- **ip-timestamp-opt** Detects packets where the IP option list includes option 4 (Internet Timestamp) and records the event in the SCREEN counters list for the ingress interface.
- **land** Prevents Land attacks by combining the SYN Flood defense mechanism with IP spoofing protection. Land attacks occur when an attacker sends spoofed IP packets with headers containing the target's IP address for both the source and destination IP addresses. The attacker sends these packets with the SYN flag set to any available port. This induces the target to create empty sessions with itself, filling its session table and overwhelming its resources.

- **limit-session** [**source-ip-based** *number* | **destination-ip-based** *number*] Limits the number of concurrent sessions the device can initiate from a single source IP address, or the number of sessions it can direct to a single destination IP address. By default, the limit is 128 sessions. Limit value range is 1 to 49,999.
- **mal-URL** [*name_str id_str number* | **code-red**] Sets up a filter that scans HTTP packets for suspect URLs. The security device drops packets that contain such URLs. The **code-red** switch enables blocking of the Code Red worm virus. Using the *name_str* option works as follows.
- *name_str* A user-defined identification name.
- *id_str* Specifies the starting pattern to search for in the HTTP packet. Typically, this starting pattern begins with the HTTP command GET, followed by at least one space, plus the beginning of a URL. (The security device treats multiple spaces between the command "GET" and the character "/" at the start of the URL as a single space.)
- *number* Specifies a minimum length for the URL before the CR-LF.
- **on-tunnel** Enables the configured screen functions on the tunnel in this zone. The default is disabled. Screening applies only to the ingress interface.

  On ASIC-based security devices, screen functions performed by the ASIC apply to tunnel traffic automatically if the functions are enabled. However, the screening is based on the physical interface the tunnel is bound to. For route based VPN tunnels, if the tunnel interface and the physical interface are in different zones, the screen functions configured for the zone of physical interface apply.
- **ping-of-death** Detects and rejects oversized and irregular ICMP packets. Although the TCP/IP specification requires a specific packet size, many ping implementations allow larger packet sizes. This can trigger a range of adverse system reactions including crashing, freezing, and restarting.
- **port-scan threshold** *number* Prevents port scan attacks. A port scan attack occurs when an attacker sends packets with different port numbers to scan available services. The attack succeeds if a port responds. To prevent this attack, the security device internally logs the number of different ports scanned from a single remote source. For example, if a remote host scans 10 ports in 0.005 seconds (equivalent to 5000 microseconds, the default threshold setting), the security device flags this as a port scan attack, and rejects further packets from the remote source. The port-scan threshold *number* value determines the threshold setting, which can be from 1000 to 1,000,000 microseconds.
- **syn-ack-ack-proxy** Prevents the SYN ACK ACK attack. Such an attach occurs when the attacker establishes multiple Telnet sessions without allowing each session to terminate. This consumes all open slots, generating a denial of service (DOS) condition.
- **syn-fin** Detects an illegal combination of flags attackers can use to consume sessions on the target device, thus resulting in a denial of service (DOS).
- **syn-flood** Detects and prevents SYN Flood attacks. Such attacks occur when the connecting host continuously sends TCP SYN requests without replying to the corresponding ACK responses.
  - **alarm-threshold** *number* Defines the number of half-complete proxy connections per second for which the security device makes entries in the event log.
  - **attack_threshold** *number* Defines the number of SYN packets per second required to trigger the SYN Proxy mechanism.

  - destination-threshold *number* Specifies the number of SYN segments received per second for a single destination IP address before the security device begins dropping connection requests to that destination. If a protected host runs multiple services, you might want to set a threshold based on destination IP address only, regardless of the destination port number.
  - drop-unknown-mac Drops packets when they contain unknown destination MAC addresses.
  - queue-size *number* Defines the number of proxy connection requests held in the proxy connection queue before the system starts rejecting new connection requests.
  - source-threshold *number* Specifies the number of SYN segments received per second from a single source IP address (regardless of the destination IP address and port number) before the security device begins dropping connection requests from that source.
  - timeout *number* Defines the maximum length of time before a half-completed connection is dropped from the queue. You can set it between 1 and 50 seconds.

- syn-frag Detects a SYN fragment attack, and drops any packet fragments used for the attack. A SYN fragment attack floods the target host with SYN packet fragments. The host caches these fragments, waiting for the remaining fragments to arrive so it can reassemble them. By flooding a server or host with connections that cannot be completed, the host's memory buffer eventually fills. No further connections are possible, and damage to the host's operating system can occur.
- tcp-no-flag Drops an illegal packet with missing or malformed flags field.
- tear-drop Blocks the Teardrop attack. Teardrop attacks occur when fragmented IP packets overlap and cause the host attempting to reassemble the packets to crash. The tear-drop option directs the security device to drop any packets that have such a discrepancy.
- udp-flood threshold *number* UDP flooding occurs when an attacker sends UDP packets to slow down the system to the point that it can no longer process valid connection requests. The threshold *number* parameter is the number of packets allowed per second to the same destination IP address/port pair. When the number of packets exceeds this value within any one-second period, the security device generates an alarm and drops subsequent packets for the remainder of that second. The valid range is from 1 to 1,000,000.
- unknown-protocol Discards all received IP frames with protocol numbers greater than 135. Such protocol numbers are undefined or reserved.
- winnuke Detects attacks on Windows NetBios communications, modifies the packet as necessary, and passes it on. (Each WinNuke attack triggers an attack log entry in the event alarm log.)

Example 1: The following command enables the ip-spoofing firewall service for the trust zone:

```
set zone trust screen ip-spoofing
```

The following command enables the ip-spoofing firewall service for the untrust zone, and instructs the device to drop any packet that has no source IP address, or that has a non-routable source IP address:

```
set zone untrust screen ip-spoofing drop-no-rpf-route
```

Example 2: The following command sets up a filter that scans HTTP packets for the code-red Code Red worm virus and drops such packets.

```
set zone untrust screen mal-url code-red
```

Example 3: The following commands block ActiveX and Java applets in HTTP traffic received on interfaces bound to the Untrust zone:

```
set zone untrust block-component activex
set zone untrust block-component java
```

Example 4: The following commands limit the number of sessions from any host in the Trust and Untrust zones to any single IP address to 80 sessions:

```
set zone trust screen limit-session destination-ip-based 80
set zone trust screen limit-session
set zone untrust screen limit-session destination-ip-based 80
set zone untrust screen limit-session
```

tcp-rst

set zone *zone* tcp-rst
unset zone *zone* tcp-rst

tcp-rst	Directs the security device to send back the TCP reset packet when it receives non-sync packets.

vrouter

set zone *zone* vrouter

vrouter	Binds the zone to a virtual router.

Creating Interfaces

Example 1: The following commands perform the following tasks:

- Create a new Layer 2 zone named L2-Marketing with VLAN ID number 1
- Assign physical interface ethernet7 to the zone
- Retrieve zone information

```
set zone name L2-Marketing L2 1
set interface ethernet7 zone L2-Marketing
```

Example 2: The following commands perform the following tasks:

- Create a new Layer 3 zone named Ext_Dept
- Bind the zone to the untrust-vr virtual router
- Enable ip-spoofing and tear-drop screening
- Bind interface ethernet4 to the zone

```
set zone name Ext_Dept
set zone Ext_Dept vrouter untrust-vr
set zone Ext_Dept screen ip-spoofing
set zone Ext_Dept screen tear-drop
set interface ethernet4 zone Ext_Dept
```

Appendix A
Interfaces

Most security zones exchange traffic with other zones (or with other devices) through physical interfaces or logical subinterfaces. Table 3 lists interface types and their descriptions.

Table 3: Interface Types

Interface Type	Description
Aggregate	aggregate*n* An aggregate interface, which is a grouping of two physical interfaces. An aggregate interface provides interface redundancy, allowing load sharing and failover.
Ethernet	ethernet*n* A physical ethernet interface, denoted by an interface port *n* and no slots. ethernet*n1*/*n2* A physical ethernet interface, denoted by an interface slot (*n1*) and a port (*n2*).
Function	mgt An interface bound to the MGT zone. ha | ha1 | ha2 The name of the dedicated HA port.
Layer 2	vlan1 The interface used for VPNs and management traffic while the device is in Transparent mode.
Loopback	loopback.*n* A logical interface that emulates a physical interface on the device. A loopback interface is always in the up state as long as the device on which it resides is up.
Redundant	redundant*n1* A redundant interface, which is a grouping of physical interfaces (each denoted by *n1*). Redundant interfaces perform interface failover. redundant*n1*.*n2* A logical redundant subinterface.
Subinterface	ethernet*n1*.*n2* A logical subinterface, denoted by an interface port (*n1*) with no slots. The .*n2* parameter identifies the logical interface. You create logical interfaces using the set interface command. ethernet*n1*/*n2*.*n3* A logical subinterface, denoted by an interface slot (*n1*) and a port (*n2*). The .*n3* parameter identifies the logical interface. You create logical interfaces using the set interface command.
Tunnel	tunnel.*n* A tunnel interface, used for VPN traffic.

Appendix B
Zones

Juniper Networks security devices use zones to host physical and logical interfaces, tunnels, and special-purpose items. Although ScreenOS has a number of default predefined zones, you can create new zones and configure them to meet the requirements of your organization. Table 4 lists zone types and their descriptions.

Table 4: Zone Types (Sheet 1 of 2)

Zone Type	Description
Layer 2	Use Layer 2 security zones when the device operates in Transparent mode. ■ **v1-trust** The V1-Trust zone, which hosts physical interfaces that communicate with trusted network space. ■ **v1-untrust** The V1-Untrust zone, which hosts physical interfaces that communicate with untrusted network space. ■ **v1-dmz** The DMZ zone, which hosts the DMZ physical interface. ■ **name** *name_str* A user-defined Layer 2 security zone. (You create such zones using the **set zone name** *name_str* L2 command.)
Layer 3	Use Layer 3 security zones when the device operates in NAT or Router mode. ■ **trust** The Trust zone, which hosts physical interfaces (and logical sub-interfaces) that communicate with trusted network space. ■ **untrust** The Untrust zone, which hosts physical interfaces (and logical sub-interfaces) that communicate with untrusted network space. ■ **global** The Global zone, which serves as a storage area for mapped IP (MIP) and virtual IP (VIP) addresses. Because traffic going to these addresses is mapped to other addresses, the Global zone does not require an interface. ■ **dmz** The DMZ zone, which hosts the DMZ physical interface. ■ **name** *name_str* A user-defined Layer 2 security zone. (You create such zones using the **set zone name** *name_str* command.)
Tunnel	Use tunnel zones to set up VPN tunnels with other security devices. ■ **untrust-tun** The Untrust-Tun zone, which hosts VPN tunnels. ■ **name** *name_str* A user-defined tunnel zone. You create such zones using the **set zone name** *name_str* **tunnel** command.

Table 4: Zone Types (Sheet 2 of 2)

Zone Type	Description
Function	Use function zones as described below: ■ null The Null zone, which serves as temporary storage for any interfaces that are not currently bound to another zone. ■ self The Self zone, which hosts the interface for remote management connections. For example, when you connect to the device via HTTP, SCS, or Telnet, you connect to the Self zone. ■ ha The HA zone, which hosts the high-availability interfaces, HA1 and HA2. ■ mgt The MGT zone, which hosts the out-of-band management interface, MGT.