ScreenOS CLI Reference Guide: IPv4 Command Descriptions

Release 5.4.0, Rev. A

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

Part Number: 530-015766-01, Revision A

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.

Writers: ScreenOS Team

Editor: Lisa Eldridge

Table of Contents

About This Guide

This guide describes the IPv4 commands used to configure and manage a security device from a console interface.

Organization

This guide includes the following sections:

- Command chapters are listed alphabetically by keyword or topic.
- Appendix A lists and briefly describes security-device interfaces.
- Appendix B lists and briefly describes zones.

Command Line Interface Syntax

Each CLI command description lists optional and mandatory dependency delimiters.

- The { and } symbols denote required keyword choices.
- The [and] symbols denote optional keyword choices. You are not required to include these choices.
- 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).

Many CLI commands have *nested* dependencies, which make features optional in some contexts and mandatory in others. For example,

[**feature_1** { **feature_2** | **feature_3** }]

In this example, the delimiters [and] surround the entire clause. You can execute the command successfully without indicating **feature_1**, **feature_2**, and **feature_3**. If you include **feature_1**, however, you must include either **feature_2** or **feature_3** because the { and } delimiters surround **feature_2** and **feature_3**.

The following example shows some of the **set interface** command's feature dependencies:

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 **arp** option's **trace-route** switch is not mandatory. The command can take any of the following forms:

set interface vlan1 broadcast flood
set interface vlan1 broadcast arp
set interface vlan1 broadcast arp trace-route

Object-Name Conventions

ScreenOS follows these conventions for object names—such as addresses, admin users, auth servers, IKE gateways, virtual systems, VPN tunnels, and zones:

- 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 key words 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 are Chinese, Korean, and Japanese.

- ASCII characters from 32 (0x20 in hexidecimal notation) 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.

Availability of 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 ?**

New, Modified, and Deleted Commands

This section lists new, modified, and deleted commands.

New Commands

The following commands are new in this release:

- cpu-limit
- dialer-pool
- dot1x
- icap
- irdp
- override
- ppp
- sm-ctx
- switch
- usb-device
- vsys-profile

Modified Commands

The following commands are modified in this release:

- alg
- anti-spam
- attack
- auth
- auth-server
- av
- counter
- di
- dns
- file
- gtp
- infranet
- interface
- modem
- performance
- policy
- port-mode
- save
- ssid
- url
- vsys
- wlan

Deleted Commands

There are no deleted commands in this release.

Juniper Networks Publications

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/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 us at the following email address:

techpubs-comments@juniper.net

access-list

Use the **access-list** commands to configure the security device for set extended access-lists to use with Policy-Based Routing (PBR).

Syntax

set

set access-list extended *ext_acl_id* [src-ip *prefix/length*] [dst-ip *prefix/length*] [src-port *min_max*] [dst-port *min_max*] [protocol *protocol*] [qos-prec *prec*] entry *acl_entry_id*

Keywords and Variables

access-list

set access-list extended *ext_acl_id* [src-ip *prefix/length*] [dst-ip *prefix/length*] [src-port *min_max*] [dst-port *min_max*] [protocol *protocol*] [qos-prec *prec*] entry *acl_entry_id*

access-list	To remove an access-list, enter **unset access-list extended entry** *acl_entry_id*.

action-group

Use the **action-group** commands to configure the security device for grouping match groups for Policy-Based Routing (PBR).

Syntax

set

set action-group *action_group_name* { [next-interface *interface_name*] [next-hop *ip_addr*] } action-entry *action_seq_number*

Keywords and Variables

action-group

set action-group *action_group_name* { [next-interface *interface_name*] [next-hop *ip_addr*] } action-entry *action_seq_number*
unset action-group *action_group_name* action-entry *action_seq_number*

action-group	Specifies the name of a match group. Each action-group name must be unique alphanumeric string and must be between 1 and 28 characters in length. An action group can specify the next interface or a next hop and associates an action-entry, which is a number between 1 and 99. The sequence number (action-entry) specifies the order in which the forwarding solution is looked for. To remove an action-group, enter **unset action-group** *action_group_name* **action-entry** *action_seq_number.*

active-user

Use the **active-user** commands to clear or display information for all users who 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

Clear

clear active-user { *IPv4 address* | all }

Get

get active-user

Keywords and Variables

all

clear active-user all

all	Deletes all active users.

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
}
[*string*]

Keywords and Variables

Variable Parameters

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 about zones, see "Zone Names" on page B-I.
fqdn	The fully-qualified domain name of the host.
ip_addr/mask	The IP address and subnet mask identifying an individual host or a subnet.
name_str	The name of the zone or group.
string	A character string containing a comment line.

Example: The following commands create address book entries named "Local_Net" and "Outside_Net":

set address trust Local_Net 10.1.1.0/24 "New_York_Subnet"
set address untrust Outside_Net 1.1.12.1/24 "London_Subnet"

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. (Create address groups using the **set group address** command.)

Example: The following command displays information for an address group named Sales_Group:

get address trust group HTTP_Servers

name

get address *zone* name *name_str*

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.

admin

Use the **admin** commands to configure or display administrative parameters for the security device. These parameters determine the following:

- Characteristics for each administrator, such as password and privilege level
- How the device performs administrator authentication
- Methods with which administrators can access the device
- An IP address or address range from which one or more administrators can connect to the device
- Which port the device uses to detect administrative traffic
- Whether the device automatically sends generated alerts and traffic alarms via email
- Whether the device is enabled for reset

Syntax

clear

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

get

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

set

set admin
{
access attempts *number* |
auth
{
banner { console login *string* | secondary *string* | telnet login *string* } |
remote { primary | read-only | root } |
server *name_str* |
timeout number |
} |
device-reset |
format { dos | unix } |
http redirect |
hw-reset |
mail
{
alert |
mail-addr1 *name_str* | mail-addr2 *name_str* |
server-name { *ip_addr* | *name_str* } |
traffic-log
} |
manager-ip *ip_addr* [*mask*] |
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 [secondary] | settings]
set admin auth banner console login *string*
set admin auth banner secondary *string*
set admin auth banner telnet login *string*
set admin auth remote { primary | read-only | root }
set admin auth server *name_str*
set admin auth timeout *number*
unset admin auth banner { console login | secondary | telnet login }
unset admin auth server
unset admin auth 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 or SSH session (**telnet**). The security device uses the banner created from the command **set admin auth banner telnet login** *string* for both Telnet and Secure Shell (SSH) logins.
  - **secondary** Specifies a second banner line that is always the same—for either console or Telnet—under the first banner line, which can be different for a console login and a Telnet login. The secondary banner can be up to 4000 bytes in length. Also, you can create an unrestricted number of line breaks by inserting the special symbol “\n” wherever you want a line to end.
- **remote { primary** | **ready-only** | **root }**
- **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.)

Example 1: The following commands create two login banners:

- “Hyperterminal Management Console“ is displayed at the start of new console admin sessions.
- “Telnet Login Here” is 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"

Example 2: The following command creates a secondary banner line with the text string "Network Empire". When an admin initiates a console or Telnet login attempt, this line will appear under the two login banners defined in the previous example:

set admin auth banner secondary "Network Empire"

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 certain 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	Enables and disables the redirection of administrative traffic to the security device from HTTP (default port 80) to HTTPS (default port 443). By default, HTTP redirection is disabled.

hw-reset

set admin hw-reset
unset admin hw-reset

hw-reset	Enables and disables 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 *name_str*

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

mail-addr2

set admin mail mail-addr2 *name_str*

mail-addr2	*name_str* 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
set admin manager-ip *ip_addr* [mask ssh [*port*]
unset admin manager-ip { *ip_addr* | all }

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. All security devices allow you to specify up to six hosts or subnets at once. **Note:** The **manager-ip** address must be unique, and different from the physical IP address of the management interface.

Example: The following command restricts management to a single host with IP address 10.1.10.100:

set admin manager-ip 10.1.10.100 255.255.255.255

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 into the device through the console only.

server-name

set admin mail server-name *ip_addr*

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 user name.
- **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 }]
set admin user name_str trustee [interface | modem]
unset admin user *name_str*

user

Creates or displays a non-root administrator (superadministrator or subadministrator). The maximum user name length is 31 characters, including all symbols except **?**. The user name is case-sensitive.

- The **privilege** switch determines the privilege level of the user (**all** or **read-only**).
- A **trustee** can be permitted to configure the untrust Ethernet interface. or the untrust modem interface. Default: none

  Admin accounts that have a trustee attribute set are restricted as follows:

  - Permitted to manage the device using the Web only.
  - Trustee accounts do not function when the device is in Transparent mode, if an account is created while the device is in Transparent mode, or when the device is in “dual-untrust” or “combined” mode.
  - Permitted only to manage a predefined set of physical interface attributes corresponding to the settings of the configured trustee attribute (interface and/or modem).

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*.

By default, HTTP redirection is *enabled* on security devices that ship with ScreenOS 5.1.0 or later.

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.

Alarm parameters determine when the device generates alarm messages along with 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 | 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 a NSRP cluster.

Example: The following command clears the alarm table entries for policy 4 and propagates the change to other device in a 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 | all }

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

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
        {
        h323 counters |
        mgcp counters |
        sccp counters |
        sip calls | counters | rate
        }
```

get

```
get alg
    {
    h323 [ counters ] |
    mgcp
      [
      calls |
      counters |
      endpoints [ name string ] |
      sessions [ dst-ip ip_addr | src-ip [ ip_addr ] ]
      ] |
    msrpc |
    rtsp |
    sip
      [
      calls [ details ] |
      counters |
      details |
      memory |
      rate |
      setting |
      transactions
      ]
    sccp
      [
      calls [ detail ] |
      counters |
      ]
    } |
    sql |
    sunrpc
    }
```

set

```
set alg
    {
    h323
      {
      app-screen
        {
        message-flood gatekeeper [ threshold number ] |
        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 ] |
    rtsp [ enable ] |
    sccp
      {
      app-screen
        {
        call flood [ threshold number ] |
        unknown message [ nat | route ] permit
        } |
      enable |
      inactive-media-timeout number
        } |
      }
    sip
      {
      C-timeout number |
      T1-interval number |
      T4-interval number |
      app-screen
        {
        protect deny [ dst-ip ip_addr/mask | timeout number ] |
        unknown-message [nat | route ] permit
        } |
      enable |
      media-inactivity-timeout number |
      signaling-inactivity-timeout number
      } |
    sql [ enable ] |
    sunrpc [ enable ]
    }
```

Keywords and Variables

h323

clear alg 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 message flood gatekeeper [threshold** *number* **]** limits the rate per second at which Remote Access Server (RAS) requests to the gatekeeper are processed. Messages exceeding **threshold** are dropped. Disabled by default. When enabled, default threshold value is 1000 connections requests; the range is 1 to 65535.
- **app-screen unknown-message [nat | route] permit** specifies how unidentified H.323 messages are handled by the security device. The default is to drop unknown (unsupported) messages. Permitting unknown messages can compromise security and is not recommended. However, in a secure test or production environmnet, this command can be useful for resolving interoperability issues with disparate vendor equipment. By permitting unknown H.323 (unsupported) messages, you can get your network operational and later analyze your VoIP traffic to determine why some messages were being dropped.

  Note that this command applies only to received packets identified as supported VoIP packets. If a packet cannot be identified, it is always dropped. If a packet is identified as a supported protocol and **unknown-message** is set to **permit**, the message is forwarded without processing.

  - **nat** specifies that unknown messages be allowed to pass if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **counters** clears all H.323 ALG counters.
- **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** *number* **]** 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 by the security device. The default is to drop unknown messages. Permitting unknown messages can compromise security and is not recommended. However, in a secure test or production environment, this command can be useful for resolving interoperability issues with disparate vendor equipment. For example, the security device rejects SIP messages containing unsupported SIP "methods." By permitting unknown SIP messages in this case, you can get your network operational and later analyze your VoIP traffic to determine why some messages were being dropped.

  Note: This command applies only to received packets identified as supported VoIP packets. If a packet cannot be identified, it is always dropped. If a packet is identified as a supported protocol and **unknown-message** is set to **permit**, the message is forwarded without processing.

  - **nat** specifies that unknown messages be allowed to pass if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **calls** displays active MGCP calls.
- **counters** displays or clears MGCP statistics.
- **enable** enables and disables the MGCP ALG (the default is enabled).
- **endpoints** displays endpoints of active sessions.
- **inactive-media-timeou**t 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) established calls are kept alive. The minimum is 3; maximum is 1440.
- **transaction-timeout** specifies the time in seconds for an MGCP transaction.The default is 30 seconds; the range is 5 to 50 seconds.

- **sessions** displays MGCP session information.
  - **dst-ip** matches the destination IP address of the session.
  - s**rc-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).

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).

sccp

clear alg sccp counters
get alg sccp [...]
set alg sccp [...]
unset alg sccp [...]

sccp

Specifies the Skinny Call Control Protocol ALG on the device.

- **app-screen call-flood [threshold** *number* **]** enables outbound call protection for the client, to protect the Call Manager from being flooded with new calls from an already compromised, connected client or a faulty device. This feature is not enabled by default. When enabled, outbound calls to Call Manager exceeding **threshold** per minute are dropped for that interval. When enabled, the default is 20 calls per minute; the range is 1 to 1000.
- **app-screen unknown-message [nat | route] permit** specifies how unidentified messages are handled by the security device. The default is to drop unknown messages. Permitting unknown messages can compromise security and is not recommended. However, in a secure test or production environment, this command can be useful for resolving interoperability issues with disparate vendor equipment. For example, the security device rejects SIP messages containing unsupported SIP "methods." By permitting unknown SIP messages in this case, you can get your network operational and later analyze your VoIP traffic to determine why some messages were being dropped.

  Note that this command applies only to received packets identified as supported VoIP packets. If a packet cannot be identified, it is always dropped. If a packet is identified as a supported protocol and **unknown-message** is set to **permit**, the message is forwarded without processing.

  - **nat** specifies that unknown messages be allowed to pass if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **calls [details]** displays the number of active calls and, optionally, information about those calls. The maximum number of calls possible on a security device depends on the platform type. For more information, refer to the specifications sheet for your product.
- **counters** displays or clears SCCP ALG statistics.
- **enable** enables and disables the SCCP ALG on the device (the default is enabled).
- **inactive-media-timeou**t *number* specifies how long pinholes and sessions opened for media are kept alive in the absence of activity. The default is 120 seconds; the range is 10 to 600 seconds.

sip

get alg sip [...]
set alg sip [...]
unset alg sip [...]
clear alg sip [...]

sip

Specifies the Session Initiation Protocol ALG on the device.

- **app-screen protect deny [dst-ip** *ip_addr/mask* | **timeout** *number* **]** specifies that repeat SIP INVITE requests be denied to a proxy server that denied the initial request.
  - **dst-ip** specifies the IP address and netmask of the proxy server or other SIP server.
  - **timeout** specifies the time in seconds the proxy server denies repeated SIP INVITE requests before it begins accepting them again. The default is 5seconds; the range is 1 to 3600 seconds.
- **app-screen unknown-message [nat** | **route] permit** specifies how unidentified messages are handled by the security device. The default is to drop unknown messages. Permitting unknown messages can compromise security and is not recommended. However, in a secure test or production environment, this command can be useful for resolving interoperability issues with disparate vendor equipment. For example, the security device rejects SIP messages containing unsupported SIP “methods.” By permitting unknown SIP messages in this case, you can get your network operational and later analyze your VoIP traffic to determine why some messages were being dropped.

  Note that this command applies only to received packets identified as supported VoIP packets. If a packet cannot be identified, it is always dropped. If a packet is identified as a supported protocol and **unknown-message** is set to **permit**, the message is forwarded without processing.
  - **nat** specifies that unknown messages be allowed to pass if the session is in NAT mode.
  - **route** specifies that unknown messages be allowed to pass if the session is in Route mode. (Sessions in Transparent mode are treated as Route mode.)
- **C-timeout** specifies the INVITE transaction timeout at the proxy, in minutes; the default is 30. Because the SIP ALG is in the middle, instead of using the INVITE transaction timer value B (which is (64 * T1) = 32 seconds), the SIP ALG gets its timer value from the proxy.
- **calls [details]** displays and clears the number of active calls and information about those calls. The maximum number of calls possible on a security device depends on the platform type. For more information, refer to the specifications sheet for your product.
- **counters** displays and clears SIP AlG statistics counters.
- **details** displays information about active calls.
- **enable** enables and disables the SIP ALG on the device (the default is enabled).
- **media-inactivity-timeout** specifies how long sessions opened are kept alive in the absence of active media. The default is 120 seconds; minimum is 10 seconds, maximum is 2550 seconds.
- **memory** displays SIP memory utilization.
- **rate** displays or clears SIP ALG performance records.

- **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. Also provides information about the SIP application screen configuration.
- **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.
- **transactions** displays SIP ALG transactions.
- **T1-interval** specifies the roundtrip time estimate, in seconds, of a transaction between endpoints. The default is 500 mseconds. Because many SIP timers scale with the T1-Interval (as described in RFC 3261), when you change the value of the T1-Interval timer, those SIP timers also are adjusted.
- **T4-interval** specifies the maximum time a message remains in the network. The default is 5 seconds. Because many SIP timers scale with the T4-Interval (as described in RFC 3261), when you change the value of the T4-Interval timer, those SIP timers also are adjusted.

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

Variable Parameters

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.

anti-spam

Use the **anti-spam** commands to create and modify an anti-spam profile. You can use these profiles in policies to filter out suspected spam messages. An anti-spam profile allows you to designate lists of IP addresses, emails, hostnames, or domain name as malicious (spam) or benign (not spam). The profile can include lists of the following types:

- Public-based whitelists or blacklists

  If the connection is from a mail-forwarding agent, the device can filter the connection's source-IP address using lists of devices deemed to be benign (whitelist) or malicious (blacklist).

- Custom-defined whitelists or blacklists

  - Domain-name-based whitelists or blacklists. The device can use such lists to filter connections that use domain names deemed to be benign or malicious.

  - Address-book-based whitelists or blacklists. The device can use such lists to base filtering on the sender's email address or domain. By default, any email server should accept its own user's email.

NOTE: This release supports anti-spam for Simple Mail Transfer Protocol (SMTP) only.

To execute most anti-spam commands, it is necessary to initiate the anti-spam context. For more information, see "Context Initiation" on page 36. This anti-spam feature is not meant to replace your anti-spam server, but to complement it.

Blacklists and Whitelists

The anti-spam feature requires that the security device have Internet connectivity with the Spam Block List (SBL) server. Domain Name System (DNS) must be available to access the SBL server. The firewall performs reverse DNS lookups on the source of the SMTP sender (or relaying agent), adding the name of the SBL server (such as sbl-server) as the authoritative domain. The DNS server then forwards each request to the SBL server, which returns a value to the device.

Alternatively, you can configure local white and blacklists. In this case, by default the system checks first against the local database of white/blacklists. If it does not find the name, the firewall proceeds to query the SBL server located on the Internet.

Basic Configuration

The following command provides a basic example of anti-spam configuration. The command is used to prevent a corporate email server from receiving and distributing spams. Corporate users retrieve emails from an internal email server without going through the firewall. This should be a typical configuration in an enterprise environment.

set anti-spam profile ns-profile
set policy from untrust to trust any mail-server SMTP permit log anti-spam ns-profile

Context Initiation

Executing the **set anti-spam profile ns-profile** command without specifying further options places the CLI within the context of a new or existing anti-spam profile. For example, you first use the following commands to define a profile named **ns-profile**, then you enter the ns-profile context to instruct the device to drop suspected spam messages:

device-> **set anti-spam profile ns-profile**
device(anti-spam:ns-profile)-> **set default action drop**

After you enter an anti-spam context, all subsequent command executions modify the specified anti-spam profile (**ns-profile** in this example). To save your changes, you must first exit the anti-spam context, then enter the **save** command:

device(anti-spam:ns-profile)-> **exit**
device-> **save**

Syntax

clear

clear anti-spam stat

exec

exec anti-spam testscan *string*

get

get anti-spam

set

set anti-spam profile ns-profile

The following **get** and **set** commands are executable in the anti-spam context.

get (within the profile context)

get { blacklist | default | sbl | whitelist }

set (within the profile context)

set
{
blacklist *string* |
default action { drop | tag [{ header | subject } string] }
sbl default-server enable |
whitelist *string*
}

Keywords and Variables

blacklist (within the profile context)

get blacklist
set blacklist *string*
unset blacklist *string*

Use the **blacklist** command to add or remove an IP address, an email, a hostname, or a domain name from the local anti-spam blacklist. Each entry in a blacklist can identify a possible spammer. The following table shows some possible entries.

Type of Entry	Sample Content
IP address	11.22.33.44
Email	admin@www.wibwaller.com
Hostname	www.wibwaller.com
Domain name	wibwaller.com

string A pattern inserted into the local blacklist. Such patterns identify spam messages. The pattern may include an IP address, an email, a hostname, or a domain name. Multiple strings are separated by semicolons (;).

Example1: These commands perform the following tasks:

1. Initiate a profile context (ns-profile).
2. Give the profile a black-list entry that prevents connections with the hostname www.wibwaller.com.
3. Exit the spam context and apply the profile to an existing policy (id 2).

device-> **set anti-spam profile ns-profile**
device(anti-spam:ns-profile)-> **set blacklist www.wibwaller.com**
device(anti-spam:ns-profile)-> **exit**
device-> **set policy id 2 anti-spam ns-profile**

Example2: These commands show blacklists with multiple entries:

device(anti-spam:ns-profile)-> **set blacklist cat@aaa.com;1.1.1.1**
device(anti-spam:ns-profile)-> **set blacklist 47.YOU2Q.COM**

default action (within the profile context)

get default
set default action drop
set default action tag header *string*
set default action tag subject *string*
unset default action

Use the **default** commands to specify how the device handles messages deemed to be spam. The device can either drop a spam message or identify it as spam by tagging it.

drop	Instructs the device to drop all messages identified as spam.
tag	Instructs the device to tag all messages identified as spam, without dropping the messages. Use *string* to tag a spam email. The default tag is ***SPAM*** and can be any user-defined string up to 40 bytes. You can place the tag in either of two email message areas: ■ **header** *string* places *string* in the header of the message. ■ **subject** *string* places *string* in the subject of the message.

Example: These commands perform the following tasks:

1. Initiate a profile context (ns-profile).
2. Specify that email messages deemed to be spam have the string "This is spam" in the message header.
3. Exit the spam context and apply the profile to an existing policy (id 2).

device-> **set anti-spam profile ns-profile**
device(anti-spam:ns-profile)-> **set default action tag header "This is spam"**
device(anti-spam:ns-profile)-> **exit**
device-> **set policy id 2 anti-spam ns-profile**

profile

set anti-spam profile *ns-profile*
unset anti-spam profile *ns-profile*

Configures the default anti-spam profile, ns-profile.

sbl (within the profile context)

get sbl
set sbl default-server-enable
unset sbl default-server-enable

Use the **sbl** command to enable use of the external spam-blocking SBL service, which uses a blacklist to identify known spam sources. The service replies to queries from the device about whether an IP address belongs to a known spammer.

default-server-enable	Enables the default SBL service. The server for this service contains a blacklist of known spam sources. The service identifies each source by an IP address.

Example: These commands perform the following tasks:

1. Initiate a profile context (ns-profile).
2. Enable use of the default anti-spam service.
3. Exit the spam context and apply the profile to an existing policy (id 2).

device-> **set anti-spam profile ns-profile**
device(anti-spam:ns-profile)-> **set sbl default-server-enable**
device(anti-spam:ns-profile)-> **exit**
device-> **set policy id 2 anti-spam ns-profile**

stat

clear anti-spam stat

Clears all accumulated statistical anti-spam counters.

testscan

exec anti-spam testscan string

Tests the anti-spam scan engine where *string* can be an IP address, a domain name, or an email address. The result is displayed to the console (serial port) only and is not displayed to a Telnet terminal. The result is also available in the debug buffer (**get dbuf stream**). Juniper Networks recommends to use this command to test your anti-spam scan engine.

Example: The following examples validate an SMTP sender. The firewall tests to see if the domain resides on the whitelist or blacklist.

exec antispam testscan spammer.org
exec antispam testscan the.very.bad.spammers.com

whitelist (within the profile context)

get whitelist
set whitelist *string*
unset whitelist *string*

Use the **whitelist** command to add or remove an IP address, an email, a hostname or a domain name from the local whitelist. Each entry in a whitelist can identify an entity that is not a suspected spammer. The following table shows some possible entries.

Type of Entry	Sample Content
IP address	11.22.33.44
Email	admin@www.wibwaller.com
Hostname	www.wibwaller.com
Domain name	wibwaller.com

string A pattern inserted into the whitelist. Such patterns identify messages that are deemed not to be spam. The pattern may include an IP address, an email, a hostname, or a domain name.

Example 1: The following two commands show a domain name and an IP address. Multiple strings are separated by semicolons (;).

```
set whitelist cat@aaa.com;1.1.1.1
set whitelist 47.YOU2Q.COM
```

Example 2: These commands perform the following tasks:

1. Initiate a profile context (ns-profile).
2. Give the profile a whitelist entry that allows connections with the hostname www.fiddwicket.com.
3. Exit the spam context and apply the profile to an existing policy (id 2).

```
device-> set anti-spam profile ns-profile
device(anti-spam:ns-profile)-> set whitelist www.fiddwicket.com
device(anti-spam:ns-profile)-> exit
device-> set policy id 2 anti-spam ns-profile
```

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

Variable Parameters

set arp *ip_addr mac_addr interface*

ip_addr	The IP address of a network device to which you want to make a static entry in the ARP table.
mac_addr	The MAC address of a network device to which you want to make a static entry in the ARP table.
interface	The name of the interface through which the security device can direct traffic to reach the network device with the specified IP and MAC addresses. For more information on interfaces, see "Interface Names" on page A-I.

all

get arp all

all	Lists all current ARP entries for every existing virtual system (vsys).

asic

get asic *id_num*

asic	Lists all current ARP entries for each Application-Specific Integrated Circuit (ASIC) chip identified by ID number.

age

set arp age *number*

age	Sets the age-out value (in seconds) for ARP entries. The default value is 1200 seconds (20 minutes).

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 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 a NetScreen Redundancy Protocol (NSRP) cluster.

attack

Use the **attack** commands to view and define attack objects, attack-object database-server settings, and download predefined signature packs.

NOTE: This command is available only if advanced mode license key is installed on the device.

Use **attack** along with the **attack-db** and **di** commands described on page 51 and page 129, respectively.

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

Variable Parameter

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	Displays protocol-anomaly attack objects currently stored in the local database.
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. See "attack-db" on page 51.

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/sigupdate
set attack db server http://services.netscreen.com/restricted/sigupdate/server
set attack db server http://services.netscreen.com/restricted/sigupdate/client
set attack db server http://services.netscreen.com/restricted/sigupdate/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 [*name_str* | 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 group, displaying—in alphabetical order—first user-defined groups (if any) and then predefined attack groups. ■ **name:** Organizes the attack-group display alphabetically by attack-group names, regardless of whether they are user-defined or predefined. However, because all user-defined attack group names must begin with “CS:”, they appear together alphabetically anyway. *name_str* specifies a name for the creation, deletion, or modification of an attack group. The keywords **add** and **remove** indicate whether you are adding or deleting an attack 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 720

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 signature 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—in alphabetical order—first user-defined objects (if any) and then predefined attack objects.
  - **id:** Organizes the stateful-signature attack-object display numerically by ID number, first listing user-defined attack objects, which have no ID number, and then predefined attack objects.
  - **name:** Organizes the stateful-signature attack-object display alphabetically by attack name.

Example: The following command displays signature-attack objects 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—first anomaly and then stateful-signature attack 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, first by anomaly and then by signature.

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

get attack sort-by id

attack-db

Use the **attack-db** commands to check and perform signature pack or attack-object database updates. ScreenOS provides four predefined signature packs. For more information on the signature packs, see "attack" on page 43. Use this **attack-db** command along with the **di** command described on page 129.

NOTE: This command is available only if Advanced mode and the Deep Inspection (DI) key are installed on the device.

Syntax

exec attack-db { check | update }

Keywords and Variables

check

exec attack-db check

check	Immediately checks if the attack-object database on the server is more recent than the one on the security device.

update

exec attack-db update

update	Updates the attack-object database on the security device immediately with the database stored on the attack-object database server.

audible-alarm

Use the **audible-alarm** commands to activate the audible-alarm feature.

Syntax

get

get audible-alarm

set

set audible-alarm { all | battery | fan-failed | power-failed | temperature }

Keywords and Variables

all

set audible-alarm all
unset audible-alarm all

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

set audible-alarm battery
unset audible-alarm battery

battery	Enables or disables the audible alarm in the event of a battery failure.

fan-failed

set audible-alarm fan-failed
unset audible-alarm fan-failed

fan-failed	Enables or disables the audible alarm in the event of a fan failure.

module-failed

set audible-alarm module-failed
unset audible-alarm module-failed

module-failed	Enables or disables the audible alarm in the event of an interface-module failure.

power-failed

set audible-alarm power-failed
unset audible-alarm power-failed

power-failed	Enables or disables the audible alarm in the event of a power-supply failure.

temperature

set audible-alarm temperature
unset audible-alarm temperature

temperature	Enables or disables the audible alarm if the temperature rises above an admin-defined threshold.

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)

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 **delete node_secret** command.

Syntax

clear

clear [cluster] auth
[
history |
queue |
statistics |
table [id *id_num* | infranet [auth_id *id_num*] | ipaddr *ip_addr*]
]

get

get auth
[
banner |
history [id *id_num* | ip *ip_addr*] |
queue |
settings [radius accounting] |
statistics |
table [id *id_num* | infranet [auth_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*
} |
}

Keywords and Variables

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 when a login attempt is unsuccessful.
  - **login** *string* Specifies a message string to display when a login prompt appears.
  - **success** *string* Specifies a message string to display when a login attempt is successful.

FTP, HTTP, and Telnet login, success, and fail banners can each be up to 4000 or greater bytes long. You can include multiple line breaks in a banner by inserting the special symbol "/n" wherever you want a line break.

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 an 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 radius accounting

settings	Displays default user-authentication server settings. (This option yields the same display as the **get auth** command.) If you specify **radius accounting**, the device displays RADIUS-related parameters.

statistics

clear auth statistics
get auth statistics

statistics	Clears or displays authentication run-time statistics.

table

clear [cluster] auth table [id *id_num* | infranet [auth_id *id_num*] | ip *ip_addr*]
get auth table [id *id_num* | infranet [auth_id *id_num*] | ip *ip_addr*]

table

Displays or clears entries in the user-authentication table. Clearing the entries forces reauthentication. 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*).
- **infranet** Clears or displays a list of all Infranet Controller authentication table entries. The output includes an auth-id, source IP address, user name, and role ID for each authentication table entry.
  - **auth_id** *id_num* Displays information about a specific Infranet Controller authentication table entry. Specify the table entry's auth-id for *id_num*. The output includes a source IP address, user name, role ID(s), and role name(s) for the table entry. (For information about how to display the role ID in the Infranet Controller, refer to the *Unified Access Control Administration Guide.*)
- **ip** *ip_addr* Clears or displays all entries with a common source-IP address (*ip_addr*).

Example 1: The following command clears entry 7 from the user authentication table:

clear auth table id 7

Example 2: 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

Example 3: The following commands display the Infranet users in the authentication table:

device-> **get auth table infranet**
```
Total Infranet users in table: 1
auth-id src user roles age status srczone dstzone
2 10.64.9.26 user1 00000000 0 N/A Null Null
```
device-> **get auth table infranet auth-id 2**
```
Infranet Auth Id: 2
Source IP: 10.64.9.26
Username: user1
Roles: 0000000001.000005.0
Roles-names: Users
User Context:
Sessions associated: 0
Zone: Null->Null
```

Note that Username, **user1** is displayed in the output of the first and second command. Roles-names, **Users** is displayed in the second output only.

auth-server

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

Syntax

get

get auth-server
  {
  name_str |
  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* |
      calling-station-id
      } |
    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* |
timeout *number* |
type { ldap | radius | securid } |
username
{
domain *dom_name* |
separator *string* number *number*
}
}

Keywords and Variables

Variable Parameter

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 { [802.1X] [admin] | [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.

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, or SecurID server type.
backup2	The IP address or DNS name of the secondary backup authentication server for an LDAP or RADIUS 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 IP address 10.1.1.50. It stores authentication user accounts. Then you define a primary backup server at 10.1.1.51 and a secondary backup server at 10.1.1.52:

set auth-server radius1 server-name 10.1.1.50
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 and LDAP 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 67.

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 explicitly, 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 { … }

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 = netscreen, 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 IP address or DNS name of the LDAP server.

Example: For an example of this option, see “Defining an LDAP Server Object” on page 69.

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 may contain the calling-station-id (described below).

  - **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*
  - **calling-station-id** Enables or disables calling-station-id transmission. The calling-station-id identifies the originator of the call. For example, this value might consist of the phone number of the user originating the call. To prevent sending this ID, disable the setting by executing the following command:
    unset auth-server *name_str* **radius attribute calling-station-id**

- **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 68.

securid

set auth-server *name_str* securid auth-port *port_num*
set auth-server *name_str* duress *number*
set auth-server *name_str* encr *id_num*
set auth-server *name_str* retries *number*
set auth-server *name_str* 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 this option, see “Defining a SecurID Server Object” on page 69.

server-name

set auth-server *name_str* server-name *ip_addr*
set auth-server *name_str* server-name *name_str*

server-name	The IP 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 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 63.

Example: For an example of this option, see "Defining a SecurID Server Object" on page 69.

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 68.

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 68.

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=netscreen;ou=marketing
save

The following command lists all auth-server settings:

get auth-server all

av

On select security devices, use the **av** commands to perform the following tasks:

- Configure your device to support an external antivirus (AV) scanner

  External AV scanning occurs when the security device redirects traffic to an external Internet Content Adaptation Protocol (ICAP) AV scan server. Use the commands in this section and in “icap” on page 213 to configure the ICAP client on your security device to support the external AV scanner.

- Configure your device to support the internal AV scanner (scan-mgr)

  Internal AV scanning occurs when the embedded scanner in the security device scans traffic for viruses. Juniper Networks supports two embedded scan engines, Trend Micro and Juniper-Kaspersky. With a few exceptions, both scan engines support all the same antivirus features.

- Support policy-based scanning

  AV scanning profiles increase the flexibility and granularity of AV scans. You may scan for viruses based on application protocol, file extensions, or content type. Profile-based scanning allows you to configure a profile to scan traffic and assign the profile to a policy.

- Download or update AV pattern files regularly for internal AV scanner

- Notify sender and receiver by email of virus information

For more information about antivirus concepts and how to use these commands, refer to *Volume 4: Attack Detection and Defense Mechanisms* in the *Concepts & Examples ScreenOS Reference Guide.*

NOTE: To activate internal AV scanning, you must first obtain and load an AV license key. An AV license is not required if you are using an external AV scanner.

Context Initiation

Executing the **set av profile** *name_str* command without specifying further options places the CLI within the context of a new or an existing AV profile. For example, the following commands initiate a custom profile named *jnpr-profile,* that by default is configured to scan FTP, HTTP, IMAP, POP3, SMTP, and ICAP traffic.

The following procedure invokes the profile and disables scanning of SMTP traffic:

1. Enter the AV profile context:

   device-> **set av profile jnpr-profile**
   device(av:jnpr-profile)->

   After you enter an AV profile context, all subsequent command executions modify the specified AV profile (*jnpr-profile*).

2. Configure the AV scan engine to disable scanning of SMTP traffic:

   device(av:jnpr-profile)-> **unset smtp enable**

3. Exit the AV profile context:

   device (av:jnpr-profile)-> **exit**

4. Link the AV profile to a firewall policy. Only one AV profile can be linked to a specific firewall policy.

   device-> **set policy id** *policy_num* **av jnpr-profile**

   For more information about assigning an AV profile to a firewall policy, see "av" on page 460.

5. Save your changes:

   device-> **save**

Syntax

clear

clear av statistics

exec (for internal av only)

exec av scan-mgr
{
pattern-download tftp-server *ip_addr* file *filename* version *number* |
pattern-update
}

get

```
get av
    {
    all |
    extension-list name_str |
    http |
    mime-list name_str |
    profile name_str |
    scan-mgr |
    session
      [ src-ip ip_addr/mask ]
        [ dst-ip ip_addr/mask ]
          [ src-port port_num1 [ port_num2 ] ]
            [ dst-port port_num1 [ port_num2 ] ] |
    statistics
    }
```

set

```
set av
    {
    all { fail-mode { traffic [ permit ] } | resources number } |
    extension-list name_str { string1[;string2...;stringn] } |
    http
      {
      keep-alive |
      trickling { number1 number2 number3 | default } |
      webmail { enable | url-pattern-name name_str { args string | host string |
        path url_str } [ exclude ] }
      } |
    mime-list name_str { string1 [;string2...;stringn ] } |
    profile name_str |
    scan-mgr
      {
      max-content-size { drop | number } |
      max-msgs { drop | number } |
      pattern-type { extended | itw | standard } |
      pattern-update-url url_str interval number |
      queue-size number
      }
    }
```

get (within a profile context)

```
get { ftp | http | imap | pop3 | smtp | icap }
```

set (within a profile context)

```
set
    {
    ftp | http | imap | pop3 | smtp | icap
      {
        decompress-layer number |
        enable |
        extension-list { include name_str | exclude name_str } |
        scan-mode
          {
          scan-all |
          scan-intelligent |
          scan-ext
          } |
        timeout number
      } |
      http skipmime { enable | mime-list string } |
      icap { name_str | req-url url_str | resp-url url_str } |
      imap | pop3 | smtp
      { email-notify
        {
        scan-error { sender | recipient } |
        virus sender
        }
      }
    }
```

Keywords and Variables

all

get av all
set av all { fail-mode traffic permit } | resources *number* }
unset av all { fail-mode traffic | resources }

all	Specifies all AV-related information, including the following: ■ **fail-mode** Determines whether traffic is permitted to pass through when an error condition occurs. The **traffic permit** switch allows the traffic to pass when an error condition occurs. ■ **resources** *number* Determines how many resources (number of connections, expressed as a percentage of total resources) the client can use. The default is 70.

Example 1: The following command allows traffic to pass when an error condition occurs:

set av all fail-mode traffic permit

Example 2: The following command instructs the device to drop traffic if an error condition occurs.This is the default behavior.

unset av all fail-mode traffic

Example 3: The following command allows each AV client to use 20 percent of the total resources:

set av all resources 20

extension-list

get av extension-list [*name_str*]
set av extension-list *name_str* { *string1* [*;string2 ...*;*stringn*] }
unset av extension-list *name_str*

extension-list	Specifies a file extension list (*name_str)* with a list of extensions (*string1* through *stringn*). The security device uses these file extensions to make decisions on which files undergo AV scanning. File extensions are case-insensitive and separated by a semicolon. An empty file extension is represented by quotation marks (“ “). The maximum length for any *name_str* is 29 bytes. The maximum length for *string1* through *stringn* is 255 bytes.

Example: The following command specifies a list named “acme” with file extensions .exe, .com, and .pdf for AV scanning.

set av extension-list acme exe;com;pdf

ftp, http, imap, pop3, smtp, icap (within a profile context)

get { ftp | http | imap | pop3 | smtp | icap }
set { ftp | http | imap | pop3 | smtp | icap } { ... }
unset { ftp | http | imap | pop3 | smtp | icap } { ... }

ftp | http | imap | pop3 | smtp | icap	Displays or sets AV scanning options for communication protocols. ■ **ftp** Enables AV scanning of File Transfer Protocol (FTP) traffic. ■ **http** Enables AV scanning of Hypertext Transfer Protocol (HTTP) traffic. ■ **imap** Enables AV scanning of Internet Mail Access Protocol (IMAP) traffic. ■ **pop3** Enables AV scanning of Post Office Protocol, version 3 (POP3) traffic. ■ **smtp** Enables AV scanning of Simple Mail Transfer Protocol (SMTP) traffic. ■ **icap** Enables external AV scanning for this profile. **Note:** External AV scanning is supported for HTTP and SMTP traffic only.

- **decompress-layer** *number* Specifies how many layers of nested compressed files the internal AV scanner can decompress before it executes the virus scan. For example, if a message contains a compressed .zip file that contains another compressed .zip file, there are two compression layers, and decompressing both files requires a **decompress-layer** setting of 2. Valid settings are between 1 and 4, so the AV scanner can decompress up to four layers of compressed files. The default for HTTP is 2; for all other protocols it is 3.

  When transmitting data, some protocols use content encoding. The AV scan engine needs to decode this layer, which is considered a decompression level before it scans for viruses.
- **enable** Enables the specified protocol.
- **extension-list { include | exclude }** Specifies the extension list (*string*) to include or exclude in the scan process. See “extension-list” on page 75. The **include** switch instructs the security device to scan the file extensions in the list. The **exclude** switch instructs the device to not scan the file extensions in the list. Only one extension list can be included or excluded for each protocol.
- **scan-mode** Specifies how the scan engine scans traffic for a specific protocol. **scan-all** specifies that the engine scan all traffic at all times. **scan-intelligent** specifies that the engine use a more sophisticated algorithm to scan the traffic. Although **scan-intelligent** is not as safe as **scan-all**, it may reduce overhead. **scan-ext** bases all scanning decisions on the file extensions in the traffic.
- **timeout** *number* Changes the timeout value for an AV session on a per-protocol basis. By default, an AV session times out after 180 seconds of inactivity. The range is 1 to 1800 seconds.

http skipmime

Skips the specified MIME list from AV scanning.

- **enable** Enables the skipmime option. By default, **skipmime** is enabled.
- **mime-list** *string* Specifies the MIME list to skip. (For more information about **mime-list**, see “mime-list” on page 79.) Only one MIME list can be linked to a profile.

imap | pop3 | smtp

email-notify Notifies the sender or recipient about detected viruses or scanning errors.

- **scan error** Sends email to **sender** or **recipient** on scanning errors.
  - **sender** Notifies **sender** if an email message is dropped as a result of a scan error.
  - **recipient** Notifies **recipient** if an email message is passed as a result of a scan error.
- **virus sender** Notifies **sender** if a virus is found in an email message.

Example: The following commands allow you to email virus or scan-error notification messages to senders or recipients. (For more information on invoking a profile, see “Context Initiation” on page 72.)

- To send virus notification messages to sender:

  device-> **set av profile jnpr-profile**
  device(av:jnpr-profile)-> **set imap email-notify virus sender**

- To send scan error notification messages to sender:

  device-> **set av profile jnpr-profile**
  device(av:jnpr-profile)-> **set imap email-notify scan-error sender**

- To disable sending scan error notification messages to recipient

  device-> **set av profile jnpr-profile**
  device(av:jnpr-profile)-> **unset imap email-notify scan-error recipient**

- To disable sending virus notification messages to sender

  device-> **set av profile jnpr-profile**
  device(av:jnpr-profile)-> **unset imap email-notify virus sender**

http

get av http
set av http { . . . }
unset av http { keep-alive | trickling | webmail { enable | url-pattern-name *name_str* } }

http — Displays or sets HTTP configuration options for AV scanning.

- **keep-alive** Directs the security device to use the HTTP keep-alive connection option. Use this option to prevent the device from modifying a connection header for each request. (By default, the device uses the HTTP close connection option.)
- **trickling** Configures the security device for HTTP trickling, which automatically forwards specified amounts of unscanned HTTP traffic to the requesting HTTP host. Trickling prevents the host from timing out while the AV scanner is busy examining downloaded HTTP files.
  - *number1* Minimum HTTP file size needed to trigger the trickling action. The default is 3 MB.
  - *number2* Size of each block of traffic the security device sends to the AV scanner. The default is 1 MB.
  - *number3* Length of each trickle of unscanned HTTP traffic that the security device forwards to the host when the conditions specified by *number2* is met. The default is 500 bytes.
  - **default** Restores all HTTP trickling settings to the default values.
- **webmail** Configures the security device for webmail scanning.
  - **enable** Enables webmail scanning only.
    The default behavior is a full HTTP scan including webmail. **Note:** Make sure a policy enabling HTTP exists.
  - **url-pattern-name** *name_str* Specifies a URL pattern name identifying a webmail type to examine for virus patterns. When the URL matches all of the following parameters, the AV scanner performs a virus scan:
    - args *string* Specifies URL arguments that begin with a “?”.
    - host *string* Specifies the host name included in the URL.
    - path url_*str* Specifies the URL path for the webmail type.

    The **exclude** switch directs the device to exclude traffic that matches any of the above specified strings.

Example 1: The following command configures HTTP trickling to trickle 800 bytes of content for every 2 MB scanned and to initiate trickling when the HTTP file is 6 MB or larger:

set av http trickling 6 2 800

Example 2: The following commands enable webmail scanning only and creates the URL pattern name "acme" for different webmail types.

- Examine for virus patterns in the host type acme.com:

  set av http webmail url-pattern-name acme host www.acme.com

- Examine for virus patterns in paths with a matching prefix string /acme/marketing:

  set av http webmail url-pattern-name acme path /acme/marketing

- Examine for virus patterns in all paths, except in paths containing the matching prefix string, /acme/marketing

  set av http webmail url-pattern-name acme path /acme/marketing exclude

- Remove the specified path type for webmail url pattern, ACME:

  unset av http webmail url-pattern-name acme path

icap (within a profile context)

get icap
unset icap
set icap { *name_str* | req-url *url_str* | resp_-rl *url_str* }
unset icap { *name_str* | req-url *url_str* | resp-url *url_str* }

icap	Displays or sets ICAP configuration options for external AV scanning
name-str	Binds a single ICAP server or an ICAP server group to the AV profile. Configures unique name strings for ICAP servers and server groups. Your security device selects either the ICAP server specified by *name-str* or the load-balanced server from an ICAP server group. The maximum string length for the server or server group name is 31 characters.
req-url	Configures the request URL string on the ICAP server to scan all POST transactions (files that are being posted to the Internet) for viruses. The default request service string, /SYMCScanReq-AV, is valid for the Symantec scan engine 5.0 ICAP server. Modify this URL string if you are communicating with a different ICAP server. The maximum string length for the URL is 255.
resp-url	Configures the response URL string on the ICAP server to scan responses returned by an HTTP/SMTP server. The default response service string, /SYMCScanResp-AV, is valid for the Symantec scan engine 5.0 ICAP server. Modify this URL string if you are communicating with a different ICAP server. The maximum string length for the URL is 255.

mime-list

get av mime-list [*name_str*]
set av mime-list *name_str { string1* [*;string2...;stringn*] }
unset av mime-list *name_str*

mime-list	Specifies a Multipurpose Internet Mail Extension (MIME) list name (*name_str)* with a list of MIME types (*string1* through *stringn*). The security device uses such MIME types to decide which HTTP traffic must undergo AV scanning. The MIME entries are case-insensitive and separated by a semicolon. An empty MIME string is invalid and should not appear in the MIME list. If the MIME entry ends with a slash (/), then the matching is a prefix match. The maximum length for *string1* through *stringn* is 40 bytes. The default MIME list, ns-skip-mime-list, includes the following predefined MIME types: ■ application/x-director ■ application/pdf ■ image/ ■ video/ ■ audio/ ■ text/css ■ text/html The maximum number of MIME lists for each vsys (and root) is 9.

Example: The following commands configure a list of HTTP MIME types (text/plain; text/css; text/html; image/) and enables the list for HTTP skipmime:

set av mime-list textmime-list text/plain;text/css;text/html;image/
set av profile HTTPProfile
device(av:HTTPProfile)-> set http skipmime enable
device(av:HTTPProfile)-> set http skipmime mime-list textmime-list

A traffic MIME type, image/gif, is a prefix match of the MIME entry image/. A traffic MIME type, text/css, is a prefix match of the MIME entry text/css. A traffic mime-type, image/gif, does not prefix-match any MIME type in the mime-list.

profile

get av profile *name_str*
set av profile *name_str*
unset av profile *name_str*

profile	Configures or displays an AV profile. Policies use AV profiles to determine which traffic undergoes AV examination and the actions to take as a result. Only one AV profile can be linked to a specific firewall policy. For more information about creating user-defined AV profiles and assigning an AV profile to a firewall policy, see "av" on page 460. Two predefined AV profiles, **ns-profile** and **scan-mgr,** exist on your device. **scan-mgr** is automatically generated during upgrade to migrate the global **scan-mgr** settings.

scan-mgr

exec av scan-mgr pattern-download tftp-server *ip_addr* file *filename* version *number*
exec av scan-mgr pattern-update
get av scan-mgr { ... }
set av scan-mgr { ... }
unset av scan-mgr { max-content-size [drop] }

scan-mgr

Configures, displays, or performs actions on parameters that control internal AV scanning:

- **max-content-size** *number* Specifies the maximum size of content for a single message that the internal AV scanner scans for virus patterns. If you enable the drop option and the total content of an incoming message exceeds the maximum, the security device drops the message content without checking for viruses. If you unset the drop option, the security device passes traffic without examining it. The range for **max-content-size** is 20 to 10,000 KB, inclusive. The default maximum content size is 10,000 KB.
- **max-msgs** *number* Specifies the maximum number of concurrent messages that the internal AV scanner scans for virus patterns. If you enable the drop option and the number of messages exceeds the maximum, the internal AV scanner drops the latest message content. The range for **max-msgs** is between 1 and 16 messages, inclusive. The default value is 16.
- **pattern-download** Retrieves pattern files directly from a host for manual updates. **Note:** This keyword is supported on the Trend Micro scan engine only.
  - **tftp-server** *ip_addr* Specifies the host from which the security device retrieves an updated pattern file.
  - **file** *filename* Specifies the name of the pattern file retrieved from a host.
  - **version** *number* Specifies the version number of the pattern file. The version number verifies the validity of the pattern file.
- **pattern-type** Selects the AV-scan engine signature databases. The selected database affects the AV scan engine's performance and coverage of virus signatures. For example, selecting the **extended** option provides a comprehensive coverage of pattern signatures but may affect the performance of the device. **Note:** This keyword is supported on the Juniper-Kaspersky scan engine only.
  - **extended** Includes virus signatures in the standard database and other supplemental databases. In addition to all virus and spyware programs, this option also detects adware, pornware, riskware, and greyware. This option may display more false positives.
  - **itw** Uses in-the-wild virus signatures only, This database detects in-the-wild virus and spyware programs. This option scans the most prevalent viruses, although it provides increased performance.
  - **standard** Uses the default standard virus database (downloaded by the **pattern-update** command), which detects all viruses (including polymorphic and other advanced viruses) and also provides inbound spyware and phishing protection.
- **pattern-update** Executes the pattern update (specified by the **pattern-update-url** option, described below).

- **pattern-update-url** *url_str* Specifies the URL address of the server from which the security device updates the pattern files. The URL address format is **http[s]://host**[:port]**/path**. (See examples below.)

  interval *number* Specifies the time interval (in minutes) between automatic updates to the signature database. Specifying a value of zero disables automatic pattern update.

- **queue-size** Determines the number of messages that each of the 16 queues can support simultaneously. After the security device sends 16 data units to the internal scanner, it stores subsequent data units in queues to await scanning. The size of each queue can range between 1 and 16. The default queue size is 16.

Example: The following commands show examples of updating pattern signatures from a URL location:

set av scan-mgr pattern-update-url http://update.juniper-updates.net/av/5gt int 60
set av scan-mgr pattern-update-url
http://5gt-p.activeupdate.trendmicro.com:80/activeupdate/server.ini int 60

session

get av session [[src-ip *ip_addr*] [dst-ip *ip_addr*] [src-port *port_num1* [*port_num2*]]
[dst-port *port_num1* [*port_num2*]]

session — Displays the status of the current application sessions and packet queue size.

- **src-ip** *ip_addr/mask* matches the source IP address and mask of the session.
- **dst-ip** *ip_addr/mask* specifies the destination IP address and mask of the session.
- **src-port** *port_num1* [*port_num2*] matches the specific source port number (lower boundary) or a range of port numbers for that session.
- **dst-port** *port_num1* [*port_num2*] matches the specific destination port number (lower boundary) or a range of port numbers for that session.

statistics

clear av statistics
get av statistics

statistics — Clears or displays all accumulated statistical AV counters.

BGP Commands

Use the **bgp** context to configure Border Gateway Protocol (BGP) in a virtual router.

Context Initiation

Initiating the **bgp** context requires the following two steps:

1. Enter the **vrouter** context by executing the **set vrouter** command:

   set vrouter *vrouter*

   where *vrouter* is the name of the virtual router. (For all examples that follow, assume that *vrouter* is the **trust-vr** virtual router.)

2. Enter the **bgp** context by executing the **set protocol bgp** command.

   device(trust-vr)-> **set protocol bgp** *as_num*

   where *as_num* is the number of the autonomous system in which the BGP routing instance resides. Once you define an autonomous system number for the BGP routing instance, you no longer have to enter the number in the **set protocol bgp** command.

BGP Command List

The following commands are executable in the **bgp** context. Click on a keyword in the table to go to complete syntax and usage information.

advertise-def-route	Use the **advertise-def-route** commands to advertise or display the default route in the current virtual router to peers. Command options: **set, unset**
aggregate	Use **aggregate** commands to create, display, or delete aggregate addresses. Aggregation is a technique for summarizing a range of routing addresses into a single route entry, expressed as an IP address and a subnet mask. Aggregates can reduce the size of the routing table, while maintaining its level of connectivity. In addition, aggregates can reduce the number of advertised addresses, thus reducing overhead. Command options: **get, set, unset**
always-compare-med	Use the **always-compare-med** commands to enable or disable the security device from comparing paths from each autonomous system (AS) using the Multi-Exit Discriminator (MED). The MED value is one of the criteria that determines the most suitable route to the neighbor device. Command options: **get, set, unset**

as-number

Use the **as-number** command to display the autonomous system number configured for the BGP routing instance. When you create the BGP routing instance in a virtual router, you must specify the autonomous system (AS) in which it resides.

Command options: **get**

as-path-access-list

Use **as-path-access-list** commands to create, remove, or display a regular expression in an AS-Path access list.

An AS-path access list serves as a packet filtering mechanism. The security device can consult such a list and permit or deny BGP packets based on the regular expressions contained in the list. The system can have up to 99 AS-path access lists.

Command options: **get, set, unset**

comm-rib-in

Use the **comm-rib-in** command to display the BGP internal routing information base learned from peers within a community.

Command options: **get**

community-list

Use **community-list** commands to enter a route in a community list, to remove a route from the list, or to display the list.

Command options: **get, set, unset**

confederation

Use the **confederation** commands to create a confederation, to remove a confederation, or to display confederation information.

Confederation is a technique for dividing an AS into smaller sub-ASs and grouping them. Using confederations reduces the number of connections inside an AS, thus simplifying full mesh topology.

Command options: **get, set, unset**

config

Use the **config** command to display the BGP configuration.

Command options: **get**

enable

Use the **enable** commands to enable or disable the BGP routing protocol in a virtual router.

Command options: **set, unset**

flap-damping

Use the **flap-damping** commands to enable or disable the flap-damping setting.

Enabling this setting blocks the advertisement of a route until the route becomes stable. Flap damping allows the security device to prevent routing instability at an AS border router, adjacent to the region where instability occurs.

Command options: **get, set, unset**

hold-time

Use the **hold-time** commands to specify or display the maximum amount of time (in seconds) that can elapse between keepalive messages received from the BGP neighbor.

Command options: **get, set, unset**

keepalive

Use the **keepalive** commands to specify the amount of time (in seconds) that elapses between keepalive packet transmissions. These transmissions ensure that the TCP connection between the local BGP router and a neighbor router stays up.

Command options: **get, set, unset**

local-pref

Use the **local-pref** command to configure a LOCAL_PREF value for the BGP routing protocol. The LOCAL_PREF attribute is the metric most often used in practice to express preferences for one set of paths over another for IBGP.

Command options: **get, set, unset**

med	Use the **med** commands to specify or display the local Multi-Exit Discriminator (MED). Command options: **get, set, unset**
neighbor	Use the **neighbor** commands to set or display configuration parameters for communicating with BGP peers. Command options: **clear, exec, get, set, unset**
network	Use the **network** commands to create, display, or delete network and subnet entries. The BGP virtual router advertises these entries to peer devices, without first requiring redistribution into BGP (as with static routing table entries). Command options: **get, set, unset**
redistribute	Use the **redistribute** commands to import routes advertised by external routers that use protocols other than BGP, or to display the current redistribution settings. Command options: **set, unset**
redistribution	Use the **redistribution** command to display the BGP redistribution rules. Command options: **get**
reflector	Use the **reflector** commands to allow the local BGP virtual router to serve as a route reflector. A *route reflector* is a router that passes Interior BGP (IBGP) learned routes to specified IBGP neighbors (*clients*), thus eliminating the need for each router in a full mesh to talk to every other router. The clients use the route reflector to readvertise routes to the entire autonomous system (AS). Command options: **get, set, unset**
reject-default-route	Use the **reject-default-route** commands to enable, disable, or display the reject-default-route setting. Enabling this setting makes the security device ignore default route advertisements from a BGP peer router. Command options: **get, set, unset**
retry-time	Use the **retry-time** command to specify the amount of time (in seconds) after failing to establish a BGP session with a peer that the local BGP routing instance retries to initiate the session. Command options: **set, unset**
rib-in	Use the **rib-in** command to display the internal routing information base learned from peers. Command options: **get**
router-id	Use the **router-id** command to display the router ID for the virtual router. Command options: **get**
synchronization	Use the **synchronization** command to enable synchronization with Interior Gateway Protocol (IGP). Command options: **set, unset**

advertise-def-route

Use the **advertise-def-route** commands to advertise or display the default route in the current virtual router to BGP peers.

Before you can execute the **advertise-def-route** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

set advertise-def-route

Keywords and Variables

None.

aggregate

Use **aggregate** commands to create, display, or delete aggregate addresses.

Aggregation is a technique for summarizing a range of routing addresses into a single route entry. Each aggregate is an address range expressed as an IP address and a subnet mask value. Aggregation can reduce the size of a router's routing table, while maintaining its level of connectivity. In addition, aggregation can reduce the number of advertised addresses, thus reducing overhead.

Before you can execute an **aggregate** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get aggregate [*ip_addr/mask*]

set

```
set aggregate
    [ ip_addr/mask [ as-set ]
      [ summary-only | suppress-map name_str ]
      [ advertise-map name_str ] [ attribute-map name_str ]
    ]
```

Keywords and Variables

advertise-map

set aggregate *ip_addr/mask* advertise-map *name_str*

advertise-map	Selects the routes that match the specified route-map for the AS-Path path attribute of the aggregate route entry.

as-set

set aggregate *ip_addr/mask* as-set [...]

as-set	Specifies that the aggregate uses an unordered set of AS numbers (the AS-Set field is set in the AS-Path path attribute) instead of an ordered sequence (the AS-Sequence field is set in the AS-Path path attribute). This option supports the aggregation of routes with different AS-Paths.

attribute-map

set aggregate *ip_addr/mask* attribute-map *name_str*

attribute-map	Changes the attributes of the aggregate route to those in the specified route map.

summary-only

set aggregate *ip_addr/mask* [as-set] summary-only

summary-only	Specifies that more specific routes that fall into the aggregate route prefix range are not advertised.

Example: The following command specifies that the aggregate uses an unordered set of AS numbers, while suppressing more specific routes.

set aggregate 3.3.3.3/24 as-set summary-only

suppress-map

set aggregate *ip_addr/mask* suppress-map *name_str*

supress-map	Suppresses the routes that match the specified route map.

always-compare-med

Use the **always-compare-med** commands to enable or disable the security device from comparing paths from each autonomous system (AS) using the Multi-Exit Discriminator (MED). The MED is one of the criteria that determines the most suitable route to the neighbor device.

Before you can execute an **always-compare-med** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get

get always-compare-med

set

set always-compare-med

Keywords and Variables

None.

as-number

Use the **as-number** command to display the autonomous system number configured for the BGP routing instance. When you create the BGP routing instance in a virtual router, you must specify the autonomous system (AS) in which it resides.

Before you can execute the **as-number** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get as-number

Keywords and Variables

None.

as-path-access-list

Use **as-path-access-list** commands to create, remove, or display a regular expression in an AS-Path access list.

An AS-path access list serves as a packet filtering mechanism. The security device can consult such a list and permit or deny BGP packets based on the regular expressions contained in the list.

Before you can execute an **as-path-access-list** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get as-path-access-list

set

set as-path-access-list *id_num* { deny | permit } *string*

Keywords and Variables

Variable Parameters

set as-path-access-list *id_num* { deny | permit } *string*
unset as-path-access-list *id_num* { deny | permit } *string*

id_num	The identification number of the access list (range 1 - 99 inclusive).
string	The regular expression used for BGP packet filtering. You can use the following in the regular expression: ■ '^' The start of a path ■ '$' The end of a path ■ '{' The start of an AS_SET ■ '}' The end of an AS_SET ■ '(' The start of an AS_CONFED_SET or AS_CONFED_SEQ ■ ')' The end of an AS_CONFED_SET or AS_CONFED_SEQ ■ '.' Matches any single character ■ '.*' Matches zero or more characters ■ '. + ' Matches one or more characters ■ '_' Matches zero or one instance of a punctuation character ■ '[]' Specifies a set of characters ■ '-' Used within brackets to specify a range of AS numbers ■ '^' Used as the first item within brackets to exclude AS numbers

deny | permit

set as-path-access-list *id_num* { deny | permit } *string*
unset as-path-access-list *id_num* { deny | permit } *string*

deny | permit	Denies or permits BGP packets containing the regular expression (*string*).

Example: The following command places the regular expression “23” in an AS-Path access list with ID number 10:

set as-path-access-list 10 permit 23

comm-rib-in

Use the **comm-rib-in** command to display the BGP internal routing information base learned from peers within a community.

Before you can execute the **comm-rib-in** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get comm-rib-in

Keywords and Variables

None.

community-list

Use **community-list** commands to create a community list that defines community attributes of routes that are permitted or denied.

A community consists of routes that are associated with the same identifier. Routers can use the community identifier when they need to treat two or more advertised routes in the same way.

Before you can execute a **community-list** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get

get community-list

set

set community-list *id_num1* { default-permit | deny | permit }
    [*number* | as *id_num2 id_num3* |
      no-advertise | no-export | no-export-subconfed | none
    ]

Keywords and Variables

Variable Parameters

set community-list *id_num1* { deny | permit | default-permit} *number*
unset community-list *id_num1* { deny | permit | default-permit} *number*

id_num1	The identifier of the community list (range 1 - 99 inclusive).

number	The community number, which can be between 0-65535 inclusive.

Example: The following command defines the community list 20 that denies routes with the community value 200.

set community-list 20 deny 200

as

set community-list *id_num1* { deny | permit } as *id_num2 id_num3*
unset community-list *id_num1* { deny | permit } as *id_num2 id_num3*

as	Defines a private community, in the form of an AS number (*id_num2*) and a community number defined within the AS (*id_num3*). The community number can be between 0-65535 inclusive.

Example: The following command creates a community list with an ID of 10 that permits the community 11 in AS 10000:

set community-list 10 permit as 10000 11

deny | permit | default-permit

set community-list *id_num1* { deny | permit } [...]
unset community-list *id_num1* { deny | permit } [...]

deny | permit	Denies or permits routes with the specified community value.
default-permit	Permits the route if it does not match any community value specified in the community list. By default, routes that do not match community values in the community list are denied.

Example: The following command defines the community list 20 that denies routes with the community value 200.

set community-list 20 deny 200

no-advertise

set community-list *id_num1* { deny | permit } no-advertise
set community-list *id_num1* { deny | permit } no-advertise

no-advertise	Specifies that the security device does not advertise routes with this community value in the communities attribute to any peer devices.

no-export

set community-list *id_num1* { deny | permit } no-export
set community-list *id_num1* { deny | permit } no-export

no-export	Specifies that the security device does not advertise routes with this community value to EBGP peers, except subautonomous systems within the confederation.

no-export-subconfed

set community-list *id_num1* { deny | permit } no-export-subconfed
set community-list *id_num1* { deny | permit } no-export-subconfed

no-export-subconfed	Specifies that the security device does not advertise routes with this community value to any external peers.

none

set community-list *id_num1* { deny | permit } none
set community-list *id_num1* { deny | permit } none

none	Specifies that the security device remove community values.

confederation

Use the **confederation** commands to create a confederation, to remove a confederation, or to display confederation information.

Confederation is a technique for dividing an AS into smaller sub-ASs and grouping them. Using confederations reduces the number of connections inside an AS, simplifying the routing matrices created by meshes.

Before you can execute a **confederation** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get confederation

set

set confederation { id *id_num1* | peer *id_num2* | rfc3065 }

Keywords and Variables

id

set confederation id *id_num1*
unset confederation id

id	The identification number (*id_num1*) of the confederation.

Example: The following command creates a confederation with an ID of 10:

set confederation id 10

peer

set confederation peer *id_num2*
unset confederation peer *id_num2*

peer *id_num2*	The identifier of a new peer autonomous system (AS) entry.

Example: The following command adds AS 45040 to the confederation:

set confederation peer 45040

rfc3065

set confederation rfc3065
unset confederation rfc3065

rfc3065	Specifies configuration in compliance with RFC 3065. The default is compliance with RFC 1965.

config

Use the **config** command to display the CLI commands used in the BGP configuration in the current virtual router.

Before you can execute the **config** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get config

Keywords and Variables

None.

enable

Use the enable commands to enable or disable the BGP routing protocol in a virtual router.

Before you can execute an **enable** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

set enable

Keywords and Variables

None.

flap-damping

Use the **flap-damping** commands to enable or disable the flap-damping setting.

Enabling this setting blocks the advertisement of a route until the route becomes stable. Flap damping allows the security device to contain routing instability at an AS border router, adjacent to the region where instability occurs.

Before you can execute a **flap-damping** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

set flap-damping

Keywords and Variables

None.

hold-time

Use the **hold-time** commands to specify or display the maximum amount of time (in seconds) that can elapse between keepalive messages received from the BGP neighbor. If the hold-time elapses before any message is received from a BGP neighbor, the session is considered down. The default is 180 seconds.

NOTE: The default keepalive value is always one-third of the current hold-time value.

Before you can execute a **hold-time** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get hold-time

set

set hold-time *number*

Keywords and Variables

Variable Parameter

set hold-time *number*

number	The maximum length of time (in seconds) between messages.

keepalive

Use the **keepalive** commands to specify the amount of time (in seconds) that elapses between keepalive packet transmissions. These transmissions ensure that the TCP connection between the local BGP router and a neighbor router stays up. The default value is one-third of the hold-time value (for the default **hold-time** value of 180 seconds, the default **keepalive** value is 60 seconds).

Before you can execute a **keepalive** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get keepalive

set

set keepalive *number*

Keywords and Variables

Variable Parameter

number	The maximum length of time (in seconds) between keepalive messages.

local-pref

Use the **local-pref** commands to configure the Local-Pref path attribute for the BGP routing protocol.

The **local-pref** path attribute is a metric used to inform IBGP peers of the local router's preference for the route. The higher the value, the greater the preference. Routers advertise this attribute to internal peers (peers in the same AS) and to neighboring confederations, but never to external peers. The default value is 100.

Before you can execute the **local-pref** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get local-pref

set

set local-pref *number*

Keywords and Variables

Variable Parameter

set local-pref *number*

number	The preference level for the virtual router.

med

Use the **med** commands to specify or display the local Multi-Exit Discriminator (MED).

MED is an attribute that notifies a neighbor in another AS of the optimal path to use when there are multiple entry points to the AS. If an EBGP update contains a MED value, the BGP routing instance sends the MED to all IBGP peers within the AS. If you assign a MED value, this value overrides any MED values received in update messages from external peers.

Although you set the MED in the local AS, the neighbor in another AS uses the MED value to decide which entry point to use. If all other factors are equal, the path with the lowest MED value is chosen. The default MED value is 0.

Before you can execute a **med** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get med

set

set med *id_num*

Keywords and Variables

Variable Parameter

set med *id_num*
unset med

id_num	The identification number of the MED.

Example: The following command specifies MED 100 for the virtual router trust-vr:

set med 100

neighbor

Use the **neighbor** commands to set or display general configuration parameters for communicating with BGP peers.

Before you can execute a **neighbor** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

clear

clear neighbor *ip_addr1*
 { flap-route *ip_addr2* [add] | soft-in | soft-out | stats }

exec

exec neighbor *ip_addr*
 { connect | disconnect | tcp-connect }

get

get neighbor { *ip_addr* | peer-group *name_str* }

set

set neighbor { *ip_addr*
  [
  advertise-def-route |
  ebgp-multihop *number* |

```
      enable |
      force-reconnect |
      hold-time number |
      keepalive number |
      md5-authentication string |
      med number |
      nhself-enable |
      peer-group name_str |
      reflector-client |
      reject-default-route |
      remote-as number
        [
        local-ip ip_addr/mask |
        outgoing-interface interface |
        src-interface interface
        ] |
      remove-private-as
      retry-time number |
      route-map name_str { in | out } |
      send-community |
      weight number
      ] |
    peer-group name_str
      [
      ebgp-multihop number |
      force-reconnect |
      hold-time number |
      keepalive number |
      md5-authentication string |
      nhself-enable |
      reflector-client |
      remote-as number |
      retry-time number |
      route-map name_str { in | out } |
      send-community |
      weight number
      ]
    }
```

Keywords and Variables

Variable Parameter

clear neighbor *ip_addr*
get neighbor *ip_addr*
set neighbor *ip_addr* { ... }
unset neighbor *ip_addr* { ... }

ip_addr	The IP address of the neighboring peer device.

Example: The following command displays information about a neighbor device at IP address 1.1.100.101:

get neighbor 1.1.100.101

advertise-def-route

set neighbor *ip_addr* advertise-def-route
unset neighbor *ip_addr* advertise-def-route

advertise-def-route	Advertises the default route in the current virtual router to the BGP peer.

connect

exec neighbor *ip_addr* connect

connect	Establishes a BGP connection to the neighbor. You can use this command for troubleshooting a BGP connection.

disconnect

exec neighbor *ip_addr* disconnect

disconnect	Terminates the BGP connection to the neighbor. You can use this command for troubleshooting a BGP connection.

ebgp-multihop

set neighbor { *ip_addr* | peer-group *name_str* } ebgp-multihop *number*
unset neighbor { *ip_addr* | peer-group *name_str* } ebgp-multihop

ebgp-multihop	The number of intervening routing nodes (*number*) allowed between the local BGP router and the BGP neighbor (*ip_addr*). A setting of zero (the default value) disables the multihop feature. The local BGP router uses the **ebgp-multihop** value as TTL in all IP packets transmitted to the neighbor.

Example: The following command directs the virtual router to allow three intervening route nodes between the virtual router and a neighbor device at IP address 1.1.100.101:

set neighbor 1.1.100.101 ebgp-multihop 3

enable

set neighbor *ip_addr* enable
unset neighbor *ip_addr* enable

enable	Enables or disables peer communications.

force-reconnect

set neighbor { *ip_addr* | peer-group *name_str* } force-reconnect
unset neighbor { *ip_addr* | peer-group *name_str* } force-reconnect

force-reconnect	Causes the peer to drop the existing BGP connection and accept a new connection. You can use this option when NSRP failover occurs but the failover interval is long enough that the BGP peer still considers the connection to be active and rejects new connection attempts.

hold-time

set neighbor { *ip_addr* | peer-group *name_str* } hold-time *number*
unset neighbor { *ip_addr* | peer-group *name_str* } hold-time

hold-time	Specifies the number of seconds (*number*) that the current BGP speaker waits to receive a message from its neighbor. The default is 180 seconds.

Example: The following command specifies a hold-time value of 60:

set neighbor 1.1.10.10 hold-time 60

keepalive

set neighbor { *ip_addr* | peer-group *name_str* } keepalive *number*
unset neighbor { *ip_addr* | peer-group *name_str* } keepalive

keepalive	Specifies the maximum amount of time (in seconds) that can elapse between keepalive packet transmissions before the local BGP virtual router terminates the connection to the neighbor. The default is one-third of the hold-time value (for the default **hold-time** value of 180 seconds, the default **keepalive** value is 60 seconds).

Example: The following command specifies a keepalive value of 90 seconds:

device(trust-vr/bgp)-> **set neighbor 1.1.100.101 keepalive 90**

md5-authentication

set neighbor { *ip_addr* | peer-group *name_str* } md5-authentication *string*
unset neighbor { *ip_addr* | peer-group *name_str* } md5-authentication *string*

md5-authentication	Specifies the BGP peer MD5 authentication string. The maximum length is 32 characters.

Example: The following command specifies an MD5 authentication string (5784ldk094):

set neighbor 1.1.100.101 md5-authentication 5784ldk094

med

set neighbor *ip_addr* med *id_num*
unset neighbor *ip_addr* med

med	Specifies the ID number (*id_num*) of the local Multi-Exit Discriminator (MED). The default value is 0.

Example: The following command specifies the Multi-Exit Discriminator (MED) 20099 for a neighbor with IP address 1.1.10.10:

set neighbor 1.1.10.10 med 20099

nhself-enable

set neighbor { *ip_addr* | peer-group *name_str* } nhself-enable
unset neighbor { *ip_addr* | peer-group *name_str* } nhself-enable

nhself-enable	Specifies that the Next-Hop path attribute for routes sent to this peer is set to the interface IP address of the local virtual router.

Example: The following command makes the local virtual router the next hop value for the peer 1.1.10.10:

set neighbor 1.1.10.10 nhself-enable

peer-group

get neighbor peer-group *name_str*
set neighbor ip_addr peer-group *name_str* [...]
set neighbor peer-group *name_str* [...]
unset neighbor ip_addr peer-group *name_str* [...]
unset neighbor peer-group *name_str* [...]

peer-group	The name of a group of BGP neighbors. Each BGP neighbor in a peer group shares the same update policies. This allows you to set up policies that apply to all the BGP peers instead of creating a separate policy for each peer. Use this command to both create the peer-group and configure peer-group parameters.

reflector-client

set neighbor { *ip_addr* | peer-group *name_str* } reflector-client
unset neighbor { *ip_addr* | peer-group *name_str* } reflector-client

reflector-client	Specifies that the neighbor is a reflector client in the route reflector cluster. The local BGP routing instance is the route reflector.

Example: The following command specifies that the neighbors in the peer group Acme_Peers are reflector clients:

set neighbor peer-group Acme_Peers reflector-client

reject-default-route

set neighbor *ip_addr* reject-default-route
unset neighbor *ip_addr* reject-default-route

reject-default-route	Specifies that the local BGP routing instance is to ignore default route advertisements from the peer. By default, default routes advertised by peers are added to the local routing table.

remote-as

set neighbor { *ip_addr* | peer-group *name_str* } remote-as *number* [local-ip *ip_addr*]
set neighbor { *ip_addr* | peer-group *name_str* }
    remote-as *number* (outgoing-interface *interface* | src-interface *interface*)
unset neighbor { *ip_addr* | peer-group *name_str* } remote-as *number* [local-ip *ip_addr*]

remote-as	Identifies the remote AS (*number*) to be the neighbor of the current BGP speaker: ■ **local-ip** *ip_addr* specifies the local IP address for EBGP multi-hop peer. ■ **outgoing-interface** *interface* specifies the outgoing interface to which BGP binds. ■ **src-interface** *interface* specifies the source interface to which the BGP binds.

Example: The following command identifies AS 30 as the remote AS for the peer 1.1.10.10:

set neighbor 1.1.10.10 remote-as 30

remove-private-as

set neighbor *ip_addr* remove-private-as
unset neighbor *ip_addr* remove-private-as

remove-private-as	Removes the private AS number from the AS-Path for this neighbor.

retry-time

set neighbor { *ip_addr* | peer-group *name_str* } retry-time *number*
unset neighbor { *ip_addr* | peer-group *name_str* } retry-time *number*

retry-time	Specifies the time (in seconds) that the BGP routing instance retries to establish a session with the peer after an unsuccessful BGP session establishment attempt. The default is 120 seconds.

route-map

set neighbor { *ip_addr* | peer-group *name_str* } route-map *name_str* { in | out }
unset neighbor { *ip_addr* | peer-group *name_str* } route-map *name_str* { in | out }

route-map	Specifies the route map to use for the BGP neighbor. The **in** | **out** switches determine if the route map applies to incoming or outgoing routes.

Example: The following command specifies that the route map Mkt_Map applies to incoming routes from the neighbor at IP address 1.1.10.10:

set neighbor 1.1.10.10 route-map Mkt_Map in

send-community

set neighbor { *ip_addr* | peer-group *name_str* } send-community
unset neighbor { *ip_addr* | peer-group *name_str* } send-community

send-community	Directs the BGP routing protocol to transmit the community attribute to the neighbor. By default, the community attribute is not sent to neighbors.

soft-in

clear neighbor *ip_addr* soft-in

soft-in	Specifies that the security device send a route-refresh request to the neighbor.

soft-out

clear neighbor *ip_addr* soft-out

soft-out	Specifies that the security device send a full routing table to the neighbor.

stats

clear neighbor *ip_addr* stats

stats	Specifies that the security device clear the neighbor's statistics.

tcp-connect

exec neighbor *ip_addr* tcp-connect

tcp-connect	Tests the TCP connection to the neighbor. You can use this command for troubleshooting a TCP connection.

weight

set neighbor { *ip_addr* | peer-group *name_str* } weight *number*
unset neighbor { *ip_addr* | peer-group *name_str* } weight

weight	The preference for routes learned from this neighbor. The higher the value, the more preference given to the routes learned from this neighbor. The default value is 100.

Example: The following command assigns a weight of 200 to the path to the neighbor at IP address 1.1.10.10:

set neighbor 1.1.10.10 weight 200

network

Use the **network** commands to create, display, or delete static network and subnet entries that are reachable from the virtual router. BGP advertises these entries to peer devices, without first requiring redistribution into BGP (as with static routing table entries).

Before you can execute a **network** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get network

set

set network *ip_addr1/mask1*
[weight number | route-map *name_str*]
[check *ip_addr2/mask2* | no-check]

Keywords and Variables

Variable Parameters

set network *ip_addr1/mask1* [...]
unset network *ip_addr1/mask1*

ip_addr1/mask1	The IP address and subnet mask of the network. The mask does not have to be the same as the subnet mask used in the network. For example, 10.0.0.0/8 is a valid network to be advertised by BGP. When the **check** option is used, ip_addr1/mask1 can be a MIP address range.

Example: The following command creates a network entry (10.1.0.0/16) for the virtual router *trust-vr*:

set network 10.1.0.0/16

check

set network *ip_addr1/mask1* check *ip_addr2/mask2*

check	Directs the device to check *ip_addr2/mask2* for network reachability before advertising *ip_addr1/mask1* to BGP peers. If *ip_addr2/mask2* is reachable, BGP advertises *ip_addr1/mask1* to its peers. If *ip_addr2/mask2* becomes unreachable, BGP withdraws the route *ip_addr1/mask1* from its peers.

no-check

set network *ip_addr1/mask1* no-check

no-check	Directs the device not to check for network reachability.

route-map

set network *ip_addr1/mask1* route-map *name_str*

route-map	Sets the attributes of this route entry to those in the specified route map.

weight

set network *ip_addr1/mask1* weight number

weight	Sets the weight of this route entry to the specified value. Enter a value between 0 and 65535.

redistribute

Use the **redistribute** commands to import routes advertised by external routers that use protocols other than BGP. Use the **get redistribution** command to display current redistribution settings.

Before you can execute a **redistribute** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get

get redistribution

set

set redistribute route-map *name_str* protocol
{ connected | imported | ospf | rip | static }

Keywords and Variables

protocol

set redistribute route-map *name_str* protocol [...]
unset redistribute route-map *name_str* protocol [...]

protocol	The protocol from which the redistributed routes were learned. This can be one of the following: connected, imported, ospf, rip, static.

route-map

set redistribute route-map *name_str* protocol [...]
unset redistribute route-map *name_str* protocol [...]

route-map	The name (*name_str*) of the route map to be used to filter routes.

redistribution

Use the **redistribution** command to display BGP redistribution rules.

Before you can execute the **redistribution** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

get redistribution

Keywords and Variables

None.

reflector

Use the **reflector** commands to allow the local virtual router to serve as a route reflector to clients in a cluster.

A *route reflector* is a router that passes Interior BGP (IBGP) learned routes to specified IBGP neighbors (*clients*), thus eliminating the need for each router in a full mesh to talk to every other router. A cluster consists of multiple routers, with a single router designated as the route reflector, and the others as clients. Routers outside of the cluster treat the entire cluster as a single entity, instead of interfacing with each individual router in full mesh. This arrangement greatly reduces overhead. The clients exchange routes with the route reflector, while the route reflector reflects routes between clients.

To configure clients in the cluster, use the reflector-client command option of neighbor on page 95.

Before you can execute a **reflector** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get

get reflector

set

set reflector [cluster-id *id_num*]

Keywords and Variables

cluster-id

set reflector cluster-id *id_num*
unset reflector cluster-id *id_num*

cluster-id	The ID number (*id_num*) of the cluster. The cluster ID allows the BGP routing instance to append the cluster ID to the cluster list of a route. BGP must be disabled before you can set the cluster ID.

Example: The following command allows the local BGP routing instance to serve as a route reflector, and sets the cluster ID to 20:

set reflector
set reflector cluster-id 20

reject-default-route

Use the **reject-default-route** commands to enable, disable, or display the reject-default-route setting. Enabling this setting makes the security device ignore default route advertisements from a BGP peer router. By default, BGP accepts default routes advertised by BGP peers.

Before you can execute an **reject-default-route** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get

get reject-default-route

set

set reject-default-route

Keywords and Variables

None.

retry-time

Use the **retry-time** command to specify the amount of time (in seconds) after failing to establish a BGP session with a peer that the local BGP routing instance retries to initiate the session. The default is 120 seconds.

Before you can execute a **retry-time** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

set retry-time *number*

Keywords and Variables

None.

rib-in

Use the **rib-in** command to display the BGP internal routing information base (RIB) learned from peers.

Before you can execute the **rib-in** command, you must initiate the **bgp** context. (See Context Initiation on page 83.)

Syntax

get rib-in [*ip_addr/mask*]

Keywords and Variables

Variable Parameter

ip_addr/mask	The network prefix for which you want to see RIB information.

router-id

Use the **router-id** command to display the router ID for the virtual router.

Before you can execute the **router-id** command, you must initiate the **bgp** context. (See “Context Initiation” on page 83.)

Syntax

get router-id

Keywords and Variables

None.

synchronization

Use the **synchronization** command to enable synchronization with an Interior Gateway Protocol (IGP), such as OSPF.

If an EBGP router advertises a route before other routers in the AS learn the route via an IGP, traffic forwarded within the AS could be dropped if it reaches a router that has not learned the route. Synchronization prevents this from occurring by ensuring that a BGP router does not advertise a route until it has also learned the route through an IGP.

Before you can execute a **synchronization** command, you must initiate the **bgp** context. (See "Context Initiation" on page 83.)

Syntax

set synchronization

Keywords and Variables

None.

chassis

Use the **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 }
```

Keywords and Variables

audible-alarm

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 a 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 or severe alarm. A severe alarm sounds a greater frequency of audible alarms and generates a greater number of event-log entries.

Example: To enable the audible alarm to sound in the event that one or more of the fans in the fan assembly fails:

set chassis audible-alarm fan-failed

clock

Use the **clock** commands to set the system time on the security device.

NOTE: By default, the security device automatically adjusts its system clock for daylight saving time.

Syntax

get

get clock

set

set clock { *date* [*time*] | dst-off | ntp | timezone *number* }

Keywords and Variables

Variable Parameters

set clock *date time*

date time	Configures the correct current date and time on the security device. Specify the date and time using the following formats: *mm/dd/yyyy hh:mm* or *mm/dd/yyyy hh:mm:ss*.

Example: The following command sets the clock to December 15, 2002, 11:00am:

set clock 12/15/2002 11:00

dst-off

set clock dst-off
unset clock dst-off

dst-off	Turns off the automatic time adjustment for daylight saving time.

ntp

set clock ntp
unset clock ntp

ntp	Configures the device for Network Time Protocol (NTP), which synchronizes computer clocks on the Internet.

timezone

set clock timezone *number*
unset clock timezone *number*

timezone	Sets the current time-zone value. This value indicates the time difference between GMT standard time and the current local time (when DST is OFF). When DST is ON and the clock is already set forward one hour, decrease the time difference by one hour and set the minutes accurately. Set the value between -12 and 12.

common-criteria

Use the **common-criteria** command 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 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 an 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 |
  hash |
  lock |
  nsmgmt-dirty |
  rollback |
  saved |
  timestamp
  ]
```

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

hash

get config hash

hash	Displays the MD5 hash of the currently running configuration.

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 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 five minutes.

nsmgmt-dirty

clear config nsmgmt-dirty
get config nsmgmt-dirty

nsmgmt-dirty — Clears the "dirty" flag, which indicates that an administrator changed a ScreenOS setting or parameter locally instead of through NSM (NetScreen-Security Manager).

ScreenOS pushes a message to NSM whenever a non-NSM entity, such as a WebUI session or a CLI-capable console session, modifies the device configuration. This message contains a flag named NSP_DEVICE_DIRECTIVE_NSMGMT_DIRTY, which informs NSM that a local change occurred. The device sends the message only once, so it does not send notice of any further locally executed changes until NSM (or a local administrator) clears the flag.

After NSM receives the message and finishes all necessary tasks in response, it issues the **clear config nsmgmt-dirty** command to the device, thus clearing the "dirty" flag.

rollback

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

rollback — Reverts the security device to the last-known-good (LKG) configuration—providing that a LKG configuration is available.

- **enable** Enables the security device to automatically roll back 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 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

timestamp

get config timestamp

timestamp — Displays the time of the latest local change made on the currently running configuration.

console

Use the **console** commands to define or list the CLI console parameters.

The console parameters determine the following:

- Whether the security device displays messages in the active console window
- The number of lines that may appear on a console window page
- The maximum time that can pass before automatic logout occurs due to inactivity

If console access is currently disabled, you can enable it using the **unset console disable** command through a Telnet connection.

Syntax

get

get console

set

set console
{
aux disable |
disable |
page *number* |
save-on-exit default-no |
timeout *number*
}

Keywords and Variables

aux disable

set console aux disable
unset console aux disable

aux disable	Enables or disables the auxiliary modem console port. Some platforms have this auxiliary port, in addition to the standard console port. An admin can use the auxiliary modem console port to execute CLI configuration commands. Use the **aux disable** switch to disable the port when you need to enforce strict security by excluding admin access through this port.

disable

set console disable

unset console disable

disable	Disables console access through the serial port. Two confirmations are required to disable access to the console. Executing this option saves the current device configuration and closes the current login session. **Note:** After you execute the **console disable** option, nonserial console sessions can still function (as with SSH and Telnet).

page

set console page *number*
unset console page

page	An integer value specifying how many lines appear on each page between page breaks. When you set this value to zero, there are no page breaks, and the text appears in a continual stream.

Example: To define 20 lines per page displayed on the console:

set console page 20

timeout

set console timeout *number*
unset console timeout

timeout	Determines how many minutes the device waits before closing an inactive administrator session. If you set the value to zero, the console never times out.

Example: To define the console timeout value to 40 minutes:

set console timeout 40

Defaults

Access to the serial console is *enabled*.

The console displays *22* lines per page.

The default inactivity timeout is *10* minutes.

The security device sends console messages to the *buffer* by default.

counter

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

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

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

Syntax

clear

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

get

```
get counter
    {
    flow | statistics
      [ interface interface [ extensive ] | 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

clear counter 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 on interfaces, see "Interface Names" on page A-I.

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 on interfaces, see "Interface Names" on page A-I.

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 on interfaces, see “Interface Names” on page A-I.

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%. 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% 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 or on a storage device.

Syntax

```
delete [ cluster ]
    {
    crypto { auth-key | file } |
    file dev_name:/filename |
    node_secret [ ipaddr ] ip_addr |
    nsmgmt keys |
    pki object-id { system | id_num } |
    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* }

file	The file residing on the module named *dev_name* from the flash card memory. Flash and USB are the only *dev_name* names available. The *filename* is the file that you want to delete that was saved on the flash card or USB storage device.

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

delete file flash:myconfig

node_secret ipaddr

delete node_secret [ipaddr] *ip_addr*

node_secret	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 { system | *id_num* }

pki obect-id	Deletes a particular PKI object, which is a four digit value (id_num) used to identify a pki object in a security device.
system	Deletes the system generated self-signed certificate.

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

di

Use the **di** commands to configure the security device to perform Deep Inspection (DI) on packets that use specified protocols.

DI is a mechanism for filtering traffic permitted by the firewall. DI enables the device to examine Layer 3 and 4 packet headers and Layer 7 application content and protocol characteristics in an effort to detect and prevent any attacks or anomalous behavior that might be present.

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

Use the **di** commands along with “attack” on page 43 and “attack-db” on page 51, respectively.

Syntax

exec

See “attack” on page 45.

get

```
get di
    {
    disable_tcp_checksum |
    service
      {
      aim
        [
        max_flap_length |
        max_icmb_length |
        max_oft_frame |
        max_tlv_length
        ] |
      dhcp
        [
        check_client_sport
        ] |
      dns
        [
        cache_size |
        cache_time |
        nxt_length |
        pointer_loop_limit |
        report_unexpected |
        report_unknowns |
        udp_message_limit
```

```
  ] |
ftp
  [
  failed_logins |
  line_length |
  password_length |
  pathname_length |
  sitestring_length |
  username_length
  ] |
gnutella
  [
  max_line_length |
  max_query_size |
  max_ttl_hops
  ] |
gopher
  [
  host_length |
  line_length
  ] |
http
  [
  alternate_ports |
  auth_length |
  brute_search |
  content_type_length |
  cookie_length |
  download_content_len number
  download_skip
  failed_logins |
  header_length |
  host_length |
  max_content_length number
  referer_length |
  request_length |
  user_agent_length
  ] |
icmp
  [
  flood_packets |
  flood_time
  ] |
ident
  [
  max_requests |
  reply_length |
  request_length
  ] |
ike
  [
  max_payloads
  ] |
imap
  [
  failed_logins |
  flag_length |
```

```
    line_length |
    literal_length |
    mbox_length |
    pass_length |
    ref_length |
    user_length
    ] |
irc
    [
    channel_length |
    nickname_length |
    password_length |
    username_length
    ] |
ldap
    [
    attributedesc_length |
    dn_max_length |
    enc_length_left_zeros |
    failed_logins |
    integer_max_bytes |
    max_mesg_size |
    mesgid_max |
    search_filter_levels |
    search_sizelimit |
    search_timelimit |
    tag_left_zeros |
    tag_max_value
    ] |
lpr
    [
    banner_length |
    cfile_length |
    cfilename_length |
    cmd_length |
    dfile_length |
    dfilename_length |
    file_format_length |
    font_length |
    mail_length |
    reply_length |
    symlink_length
    ] |
msn
    [
    max_display_name |
    max_group_name |
    max_ip_port |
    max_phone_number |
    max_url |
    max_user_name |
    max_user_state
    ] |
msrpc
    [
    epm_max_num_entries |
    epm_max_tower_len |
```

```
    max_frag_len |
    ] |
nbname
    [
    pointer_loop_limit |
    ] |
nfs
    [
    max_buffer_length |
    max_name_length |
    max_path_length
    ] |
ntp
    [
    ctl_auth_len |
    dmsg_ver3_max_len |
    dmsg_ver4_max_len |
    match_ts |
    max_clkage |
    max_data_store |
    max_stratum |
    min_poll |
    pasv_dissolve_tm |
    varname_len |
    varvalue_len |
    ] |
pop3
    [
    apop_length |
    failed_logins |
    line_length |
    max_msg_num |
    pass_length |
    user_length
    ] |
radius
    [
    failed_auth
    ] |
smb
    [
    failed_logins |
    regkey_length |
    ] |
smtp
    [
    check_headers_in_body |
    cmdline_length |
    content_filename_length |
    content_name_length |
    domain_length |
    multipart_depth |
    num_rcpt |
    parse_cnt_length |
    path_length |
    replyline_length |
    textline_length |
```

```
        user_length
        ] |
    syslog
        [
        validate_timestamp
        ] |
    telnet
        [
        failed_logins
        ] |
    tftp
        [
        filename_length
        ] |
    vnc
        [
        failed_logins |
        max_cuttext_length |
        max_name_length |
        max_reason_length |
        verify_message
        ] |
    whois
        [
        request_length
        ] |
    ymsg
        [
        max_activity |
        max_buddy_list |
        max_challenge |
        max_chatroom_msg |
        max_chatroom_name |
        max_conf_msg |
        max_conference_name |
        max_cookie_length |
        max_crypt |
        max_file_name |
        max_group_name |
        max_mail_address |
        max_mail_subject |
        max_message_size |
        max_url_name |
        max_user_name |
        max_webcam_key |
        max_yahoo_message
        ]
    }
}
```

set

```
set di
    {
    disable_tcp_checksum |
    service
      {
      aim
        {
        max_flap_length number |
        max_icmb_length number |
        max_oft_frame number |
        max_tlv_length number
        } |
      dhcp
        {
        check_client_sport number
        } |
      dns
        {
        cache_size number |
        cache_time number |
        nxt_length number |
        pointer_loop_limit number |
        report_unexpected number |
        report_unknowns number |
        udp_message_limit number
        } |
      ftp
        {
        failed_logins number |
        line_length number |
        password_length number |
        pathname_length number |
        sitestring_length number |
        username_length number
        } |
    gnutella
        {
        max_line_length number |
        max_query_size number |
        max_ttl_hops number
        } |
    gopher
        {
        host_length number |
        line_length number
        } |
    http
        {
        alternate_ports number |
        auth_length number |
        brute_search number |
        content_type_length number |
        cookie_length number |
        download_content_len number
        download_skip
```

failed_logins |
header_length |
host_length |
max_content_length *number*
referer_length *number* |
request_length *number* |
user_agent_length *number*
} |
icmp
{
flood_packets *number* |
flood_time *number*
} |
ident
{
max_requests *number* |
reply_length *number* |
request_length *number*
} |
ike
{
max_payloads *number*
} |
imap
{
failed_logins *number* |
flag_length *number* |
line_length *number* |
literal_length *number* |
mbox_length *number* |
pass_length *number* |
ref_length *number* |
user_length *number*
} |
irc
{
channel_length *number* |
nickname_length *number* |
password_length *number* |
username_length *number*
} |
ldap
{
attributedesc_length *number* |
dn_max_length *number* |
enc_length_left_zeros *number* |
failed_logins *number* |
integer_max_bytes *number* |
max_mesg_size *number* |
mesgid_max *number* |
search_filter_levels *number* |
search_sizelimit *number* |
search_timelimit *number* |
tag_left_zeros *number* |
tag_max_value *number*
} |
lpr

{
banner_length *number* |
cfile_length *number* |
cfilename_length *number* |
cmd_length *number* |
dfile_length *number* |
dfilename_length *number* |
file_format_length *number* |
font_length *number* |
mail_length *number* |
reply_length *number* |
symlink_length *number*
} |
msn
{
max_display_name *number* |
max_group_name *number* |
max_ip_port *number* |
max_phone_number *number* |
max_url *number* |
max_user_name *number* |
max_user_state *number*
} |
msrpc
{
epm_max_num_entries *number* |
epm_max_tower_len *number* |
max_frag_len *number* |
} |
nbname
{
pointer_loop_limit *number* |
} |
nfs
{
max_buffer_length *number* |
max_name_length *number* |
max_path_length *number*
} |
ntp
{
ctl_auth_len *number* |
dmsg_ver3_max_len *number* |
dmsg_ver4_max_len *number* |
match_ts *number* |
max_clkage *number* |
max_data_store *number* |
max_stratum *number* |
min_poll *number* |
pasv_dissolve_tm *number* |
varname_len *number* |
varvalue_len *number* |
} |
pop3
{
apop_length *number* |
failed_logins *number* |

```
    line_length number |
    max_msg_num number |
    pass_length number |
    user_length number
    } |
radius
    {
    failed_auth
    } |
smb
    {
    failed_logins number |
    regkey_length number
    } |
smtp
    {
    check_headers_in_body number |
    cmdline_length number |
    content_filename_length number |
    content_name_length number |
    domain_length number |
    multipart_depth number |
    num_rcpt number |
    parse_cnt_length number |
    path_length number |
    replyline_length number |
    textline_length number |
    user_length number
    }
syslog
    {
    validate_timestamp number
    } |
telnet
    {
    failed_logins number
    } |
tftp
    {
    filename_length number
    } |
vnc
    {
    failed_logins number |
    max_cuttext_length number |
    max_name_length number |
    max_reason_length number |
    verify_message number
    } |
whois
    {
    request_length number
    } |
ymsg
    {
    max_activity number |
    max_buddy_list number |
```

```
            max_challenge number |
            max_chatroom_msg number |
            max_chatroom_name number |
            max_conf_msg number |
            max_conference_name number |
            max_cookie_length number |
            max_crypt number |
            max_file_name number |
            max_group_name number |
            max_mail_address number |
            max_mail_subject number |
            max_message_size number |
            max_url_name number |
            max_user_name number |
            max_webcam_key number |
            max_yahoo_message number
            }
        }
    }
```

Keywords and Variables

disable_tcp_checksum

get disable_tcp_checksum
set disable_tcp_checksum
unset disable_tcp_checksum

disable_tcp_checksum	Disables the TCP-checksum operation. The security device uses TCP checksums in exchanged packets to detect TCP transmission errors. Because the checksum operation uses up processor resources, it may be useful to disable it. The security device performs the checksum operation by default.

Example 1: The following command disables the checksum operation:

set di disable_tcp_checksum

Example 2: The following command enables the checksum operation:

unset di disable_tcp_checksum

aim

get di service aim { ... }
set di service aim { ... }
unset di service aim { ... }

aim

Determines how the security device evaluates America Online Instant Messaging (AIM) traffic. AIM makes use of the Open System for Communication in Real Time (OSCAR) protocol, which in turn uses FDDITalk Link Access Protocol (FLAP) for packet structuring.

- **max_flap_length** *number* Specifies the maximum number of bytes in a FLAP packet—6-byte header + data.

  Valid range: 6 - 10,000 bytes; default: 10,000 bytes.

- **max_icmb_length** *number* Specifies the maximum number of bytes in an inter-client-message block (ICMB). When an instant message is transmitted, the FLAP protocol breaks it into multiple ICMBs and sends each block in a separate Type, Length, and Value (TLV).

  Valid range: 0 - 10,000 bytes; default: 2000 bytes.

- **max_oft_frame** *number* Specifies the maximum number of bytes in an OSCAR file transfer (OFT) frame.

  Valid range: 0 - 10,000 bytes; default: 10,000 bytes.

- **max_tlv_length** *number* Specifies the length of a TLV unit. A TLV unit consists of a 2-byte type code + a 2-byte value for Length + the actual data in the Value field. TLVs often appear in the FLAP data field.

  Valid range: 0 - 100,000; default: 8000.

dhcp

get di service dhcp { check_client_sport }
set di service dhcp { check_client_sport }
unset di service dhcp { check_client_sport }

dhcp

check-client-sport { 0 | 1 } allows you to set the device to verify that the client's source port is 68. This feature is disabled by default (0). Set the value to 1 to enable this option.

dns

get di service dns { ... }
set di service dns { ... }
unset di service dns { ... }

dns

Determines how the security device evaluates Domain Name System (DNS) traffic and how it caches DNS queries.

- **cache_size** *number* The maximum size, in bytes, of the DNS cache on the security device.

  Valid range: 0 - 1,000,000 ; default: 100.

- **cache_time** *number* The maximum number of seconds that the security device stores a query in its cache.

  Valid range: 0 - 3600 ; default: 60.

- **nxt_length** *number* The maximum number of bytes in a nonexistent resource record (NXT RR) in a DNS response message.

  Valid range: 1024 - 8192; default: 4096.

- **pointer_loop_limit** *number* The valid range is 0 through 24; default: 8.
- **report_unexpected { 0 | 1 }** Enables or disables the reporting of unexpected DNS parameters. A value of 0 disables such reporting, and 1 enables it. The following are examples of unexpected DNS parameters:
  - The TYPE value is equal to or greater than 252. Values equal to and greater than 252 are reserved for QTYPE fields. (Refer to RFC 1035, *Domain Names – Implementation and Specification*.)
  - The RR TYPE code is 249, but the CLASS code is not 255 (any class). TYPE 249 is for the Transaction Key (TKEY) RR. The TKEY RR provides a mechanism with which a DNS server and resolver can establish shared secret keys to authenticate the DNS queries and responses passing between them. (Refer to RFC 2930, *Secret Key Establishment for DNS (TKEY RR)*.)

By default, the reporting of unexpected DNS parameters is disabled.

- **report_unknowns { 0 | 1 }** Enables or disables the reporting of any unknown DNS TYPE and CLASS parameter. A value of 0 disables such reporting, and 1 enables it. An unknown DNS TYPE or CLASS is anything not defined in one of the following DNS-related RFCs: 1035, 1183, 2535, 1712, 1876, 1886, 1995, 2053, 2065, 2538, 2671, 2672, and 2930. By default, the reporting of unknown DNS parameters is disabled.
- **udp_message_limit** *number* Specifies the maximum number of bytes in a UDP message sent during a DNS exchange.

  Valid range: 512 - 4096; default: 512.

ftp

get di service ftp { ... }
set di service ftp { ... }
unset di service ftp { ... }

ftp

Determines how the security device evaluates File Transfer Protocol (FTP) traffic. The security device compares actual FTP traffic with maximum settings of what you consider to be normal FTP traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **failed_logins** *number* Specifies the maximum number of failed login attempts per minute to an FTP server from a single host.

  Valid range: 2 - 100; default: 8.
- **line_length** *number* Specifies the maximum number of bytes in an FTP command line.

  Valid range: 1 - 8192; default: 1024.
- **password_length** *number* Specifies the maximum number of bytes for an FTP password.

  Valid range: 1 - 8192; default: 64.
- **pathname_length** *number* Specifies the maximum number of bytes in an FTP path name.

  Valid range: 1 - 8192; default: 512.

  sitestring_length *number* Specifies the maximum number of bytes in an FTP site string.
  Valid range: 1 - 8192; default: 512.

  username_length *number* Specifies the maximum number of bytes in an FTP username.
  Valid range: 1 - 8192; default: 32.

gnutella

get di service gnutella { ... }
set di service gnutella { ... }
unset di service gnutella { ... }

gnutella

Determines how the security device evaluates Gnutella traffic. Gnutella is a peer-to-peer (P2P) file-sharing protocol and application that does not make use of centralized servers.

- **max_line_length** *number* Specifies the maximum number of bytes in a Gnutella command line.

  Valid range: 1 - 4096; default: 2048.
- **max_query_size** *number* Specifies the maximum number of bytes in a query sent between two Gnutella peers.

  Valid range: 256 - 4096; default: 256.
- **max_ttl_hops** *number* Specifies the maximum number of network forwarding devices (hops) already passed plus the remaining Time to Live (TTL) value indicated in the Gnutella header.
  Valid range: 1 - 10; default: 8.

gopher

get di service gopher [...]
set di service gopher { ... }
unset di service gopher { ... }

gopher

- **host_length** Specifies the maximum length of the host name.

  Valid range: 1 through 128; default: 64.
- **line_length** Specifies the maximum number of lines.

  Valid range: 1 through 2048. default: 512.

http

get di service http { ... }
set di service http { ... }
unset di service http { ... }

http

Determines how the security device evaluates HyperText Transfer Protocol (HTTP) traffic. The security device compares actual HTTP traffic with maximum settings of what you consider to be normal HTTP traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **alternate_ports { 0 | 1 }** Enables or disables the inspection of HTTP traffic on the default HTTP port of 80 and on the following ports: 7001, 8000, 8001, 8100, 8200, 8080, 8888, and 9080. A value of 0 disables HTTP traffic inspection on these alternative ports, and 1 enables it. By default, this option is enabled.
- **auth_length** *number* Specifies the maximum number of bytes in an HTTP header-authorization line.

  Valid range: 1 - 1024; default: 512.
- **brute_search** *number* Specifies the maximum number of HTTP errors per minute. If the security device detects more HTTP 301 (Moved Permanently), 403 (Forbidden), 404 (Not Found), and 405 (Method Not Allowed) errors than the specified maximum, the device considers it an anomalous event.

  Valid range: 2 - 100; default: 16.
- **content_type_length** *number* Specifies the maximum number of bytes for an HTTP header Content Type field, which specifies the media type of the data contained in the HTTP packet.

  Valid range: 1 - 8192; default: 512.
- **cookie_length** *number* Specifies the maximum number of bytes in a cookie.

  Valid range: 1 - 8192; default: 8192.

  Cookies that exceed the cookie-length setting can match the protocol anomaly HTTP-HEADER-OVERFLOW and produce unnecessary log records. If the security device generates too many log records for this anomaly, increase the cookie-length setting.
- **download_content_len** *number* Specifies the maximum number of bytes of HTTP downloads.

  Valid range: 0 - 2GB; default: 2GB.
- **download-skip** Skips checking HTTP downloads for attacks. This is the default. Use the **unset** command to always check HTTP downloads for attacks.
- **failed_logins** *number* Specifies the maximum number of failed login attempts per minute to an HTTP server from a single host.

  Valid range: 2 - 100; default: 8.
- **header_length** *number* Specifies the maximum number of bytes for an HTTP packet header.

  Valid range: 1 - 8192; default: 8192.
- **host_length** *number* Specifies the maximum number of bytes for an HTTP header host, which can be an Internet domain name or an IP address.

  Valid range: 1 - 8192; default: 64.

- **max_content_length** *number* Specifies the maximum number of bytes of text or HTML content that is downloaded.

  Valid range: 0 - 2GB; default: less than 2GB.

- **referer_length** *number* Specifies the maximum number of bytes for a header-referer field, which the client uses to specify the address Uniform Resource Identifier (URI), which is a formatted string that identifies a network resource by a characteristic such as a name or a location.

  Valid range: 1 - 8192; default: 8192

- **request_length** *number* Specifies the maximum number of bytes for an HTTP request, which includes information such as a network-resource identifier, the method to apply to the resource, and the protocol version.

  Valid range: 1 - 8192; default: 8192

  user_agent_length *number* Specifies the maximum number of bytes for an HTTP header user-agent field, which contains information about the user agent that originated the request.

  Valid range: 1 - 8192; default: 256

icmp

get di service icmp [...]
set di service icmp { ... }
unset di service icmp { ... }

icmp

- **flood_packets** *number* Specifies the maximum number of packets per second to trigger a flood. Valid range: 1 through 65535; default: 250.
- **flood_time** *number* Specifies the minimum number of seconds between packets. Valid range: 1 through 65535; default: 1.

ident

get di service ident [...]
set di service ident { ... }
unset di service ident { ... }

ident

- **max_requests** *number* Specifies the maximum number of requests per session. Valid range: 1 through 65535; default: 1.
- **reply_length** *number* Specifies the maximum length of a reply. Valid range: 1 through 8192; default: 128.
- **request_length** *number* Specifies the maximum length of a request length. Valid range: 1 through 8192; default: 15.

ike

get di service ike [...]
set di service ike max_payloads
unset di service ike max_payloads

ike

max_payloads *number* Valid range: 1 through 256; default: 57.

imap

get di service imap { ... }
set di service imap { ... }
unset di service imap { ... }

imap

Determines how the security device evaluates Internet Message Access Protocol (IMAP) traffic. The security device compares actual IMAP traffic with maximum settings of what you consider to be normal IMAP traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **failed_logins** *number* Specifies the maximum number of failed login attempts per minute to an IMAP server from a single host.

  Valid range: 2 - 100; default: 8.
- **flag_length** *number* Specifies the maximum number of bytes for an IMAP flag.

  Valid range: 1 - 8192; default: 64.
- **line_length** *number* Specifies the maximum number of bytes for an IMAP line.

  Valid range: 1 - 8192; default: 2048.
- **literal_length** *number* Specifies the maximum number of octets in a literal string. In IMAP4, a string can be in one of two forms: literal or quoted. As defined in RFC 2060, *Internet Message Access Protocol – Version 4rev1*:

  A literal is a sequence of zero or more octets (including CR and LF), prefix-quoted with an octet count in the form of an open brace ("{"), the number of octets, close brace ("}"), and CRLF.

  Valid range: 1 - 16,777,215; default: 65,535.
- **mbox_length** *number* Specifies the maximum number of bytes for an IMAP mailbox.

  Valid range: 1 - 8192; default: 64.
- **pass_length** *number* Specifies the maximum number of bytes for an IMAP password.

  Valid range: 1 - 8192; default: 64.
- **ref_length** *number* Specifies the maximum number of bytes for an IMAP reference.

  Valid range: 1 - 8192; default: 64.
- **user_length** *number* Specifies the maximum number of bytes for an IMAP username.

  Valid range: 1 - 8192; default: 64.

irc

get di service irc [...]
set di service irc { ... }
unset di service irc { ... }

irc	■ **channel_length** *number* Specifies the maximum channel length. Valid range: 1 through 512; default: 64. ■ **nickname_length** *number* Specifies the maximum length for a nickname. Valid range: 1 through 512; default: 16. ■ **password_length** *number* Specifies the maximum length for a password. Valid range: 1 through 512; default: 16. ■ **username_length** *number* Specifies the maximum length for a username. Valid range: 1 through 512; default: 16.

ldap

get di service ldap [...]
set di service ldap { ... }
unset di service ldap { ... }

ldap	■ **attributedesc_length** *number* Specifies the maximum length of the attribute descriptor. Valid range: 0 through 4096; default: 512. ■ **dn_max_length** *number* Specifies the maximum length for an LDAP distinguished name. Valid range: 0 through 4096; default: 512. ■ **enc_length_left_zeros** *number* Specifies the number of left zeros for the length of the BER. Valid range: 0 through 1024; default: 64. ■ **failed_logins** *number* Specifies the maximum number of failed logins per minute. Valid range: 2 through 100; default: 8. ■ **integer_max_bytes** *number* Specifies the maximum length of integer representation in BER. Valid range: 0 through 1024; default: 4. ■ **max_mesg_size** *number* Specifies the maximum size of an LDAP message. Valid range: 0 through 8192; default: 8100. ■ **mesgid_max** *number* Specifies the maximum size of an LDAP message ID. Valid range: 0 through 2,147,483,647; default: 2,147,483,647. ■ **search_filter_levels** *number* Specifies the maximum number of nested operators in a search request. Valid range: 1 through 100; default: 8. ■ **search_sizelimit** *number* Specifies the maximum number of search results requested. Valid range: 0 through 2,147,483,647; default: 0. ■ **search_timelimit** *number* Specifies the maximum amount of time to search results requested. Valid range: 0 through 600,000; default: 0. ■ **tag_left_zeros** *number* Specifies the number of left zeros for a tag in the BER. Valid range: 0 through 1024; default: 4. ■ **tag_max_value** *number* Specifies the maximum value for any LDAP tag in the BER. Valid range: 0 through 31; default: 31.

lpr

get di service lpr [...]
set di service lpr { ... }
unset di service lpr { ... }

lpr

- **banner_length** *number* Specifies the maximum length of the banner.

  Valid range: 1 through 1024; default: 32.
- **cfile_length** *number* Specifies the maximum value of the control file size.

  Valid range: 1 through 4,294,967,295; default: 1024.
- **cfilename_length** *number* Specifies the maximum length of the control filename.

  Valid range: 1 through 1024; default: 64.
- **cmd_length** *number* Specifies the maximum subcommand length of the RECEIVE-JOB command.

  Valid range: 1 through 8192; default: 256.
- **dfile_length** *number* Specifies the maximum data-file size.

  Valid range: 1 through 4,294,967,295; default: 65535.
- **dfilename_length** *number* Specifies the maximum length of a data filename.

  Valid range: 1 through 1024; default: 64.
- **file_format_length** *number* Specifies the maximum filename length of format-related subcommands.

  Valid range: 1 through 1024; default: 32.
- **font_length** *number* Specifies the maximum font length.

  Valid range: 1 through 1024; default: 64.
- **mail_length** *number* Specifies the maximum size of an email message.

  Valid range: 1 through 1024; default: 32.
- **reply_length** *number* Specifies the maximum length of a reply from the server.

  Valid range: 1 through 8192; default: 256.
- **symlink_length** *number* Specifies the maximum symbolic length.

  Valid range: 1 through 1024; default 1024.

msn

get di service msn { ... }
set di service msn { ... }
unset di service msn { ... }

msn

Determines how the security device evaluates Microsoft Network Instant Messaging (MSN IM) traffic. The security device compares actual MSN traffic with maximum settings of what you consider to be normal MSN traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **max_display_name** *number* Specifies the maximum number of bytes in an MSN display name, which is the name that you use to identify yourself to other MSN principals. A display name is also known as a *friendly name*, *custom name* or *custom username*.

  Valid range: 1 - 1024; default: 128.

- **max_group_name** *number* Specifies the maximum number of bytes for an MSN group. Every group has a name and an ID number, and every principal belongs to at least one group: the default group named " ~ " (tilde) with ID 0.

  Valid range: 1 - 1024; default: 84.

- **max_ip_port** *number* Specifies the maximum number of bytes for the IP address:port number of an MSN server (notification or switchboard server) for a switchboard session.

  Valid range: 30 - 40; default: 30.

  All MSN notification and switchboard servers use port 1863.

- **max_phone_number** *number* Specifies the maximum number of bytes for a telephone number in an MSN Forward List (FL). The FL is essentially a contact list of other MSN principals.

  Valid range: 20 - 50; default: 20.

- **max_url** *number* Specifies the maximum number of bytes for a URL address in an MSN message.

  Valid range: 1 - 2000; default: 1024.

- **max_user_name** *number* Specifies the maximum number of bytes in any MSN user's name.

  Valid range: 1 - 1024; default: 84.

- **max_user_state** *number* Specifies the maximum number of bytes in an MSN user state, which is a 3-letter code that indicates the status of a user's connection. Some examples: NLN (online), FLN (offline), HDN (hidden/invisible). Other states are substates of NLN, including BSY (Busy), IDL (Idle), and BRB (Be Right Back).

  Valid range: 3 - 15; default: 3.

msrpc

```
get di service msrpc { ... }
set di service msrpc { ... }
unset di service msrpc { ... }
```

msrpc — Determines how the security device evaluates Microsoft Remote Procedure Call (MSRPC) traffic. The security device compares actual MSRPC traffic with maximum settings of what you consider to be normal MSRPC traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **epm_max_num_entries** *number* Specifies the maximum number of entries in an MSRPC endpoint mapper (EPM) message.

  Valid range: 100 - 8192; default: 100.

- **epm_max_tower_len** *number* Specifies the maximum number of bytes in a protocol-tower representation in an MSRPC EPM message. A protocol tower consists of an interface identifier and binding information between a client and server that permits the client to make a remote procedure call to the server.

  Valid range: 8192 - 268,435,456; default: 8192.

- **max_frag_len** *number* Specifies the maximum length, in bytes, of an MSRPC fragment.

  Valid range: 4096 - 65,535; default: 8192.

nbname

get di service nbname { ... }
set di service nbname { ... }
unset di service nbname { ... }

nbname	Determines how the security device evaluates NetBIOS name (Nbname) traffic. The security device compares actual Nbname traffic with maximum settings of what you consider to be normal Nbname traffic. The security device considers any traffic exceeding such settings to be anomalous. ■ **pointer_loop_limit** *number* Specifies the maximum number of pointer-loop levels for NetBIOS names. Valid range: 0 - 24; default: 8.

nfs

get di service nfs [...]
set di service nfs { ... }
unset di service nfs { ... }

nfs	■ **max_buffer_length** *number* Specifies the maximum buffer size for read/write requests. Valid range: 1 through 65536; default: 32768. ■ **max_name_length** *number* Specifies the maximum length for the name. Valid range: 1 through 4096; default; 256. ■ **max_path_length** *number* Specifies the maximum value for the path length. Valid range: 1 through 4096; default: 1024.

ntp

get di service ntp [...]
set di service ntp { ... }
unset di service ntp { ... }

ntp	Determines how the security device evaluates Network TIme Protocol (NTP) traffic.

- **ctl_auth_len** *number* Specifies the maximum size of the authentication-field length in the control message.

  Valid range: 0 through 24; default: 20.
- **dmsg_ver3_max_len** *number* Specifies the maximum length of an NTP version 3 message.

  Valid range: 0 through 72; default: 68.
- **dmsg_ver4_max_len** *number* Specifies the maximum length of an NTP version 4 message.

  Valid range: 0 through 72; default: 68.
- **match_ts { 0 | 1 }** Enables (1) or disables (0) the feature that matches the timestamps of NTP requests and responses. Default: 1.
- **max_clkage** *number* Specifies the maximum time since the last update of the reference clock.

  Valid range: 0 through 86400; default: 86400.
- **max_data_store** *number* Specifies the maximum buffer length to store between control packets.

  Valid range: 0 through 255; default: 255.
- **max_stratum** *number* Specifies the maximum stratum value for any NTP peer. Valid range: 0 through 15; default: 15.
- **min_poll** *number* Specifies the minimum number of seconds between two requests.

  Valid range: 0 through 1024; default: 0.
- **pasv_dissolve_tm** *number* Specifies the maximum time for a symmetric passive association to dissolve.

  Valid range: 0 through 3600; default: 900.
- **varname_len** *number* Specifies the maximum length of any NTP control variable.

  Valid range: 0 through 255; default: 128.
- **varvalue_len** *number* Specifies the maximum length of any NTP variable.

  Valid range: 0 through 255; default: 255.

pop3

get di service pop3 { ... }
set di service pop3 { ... }
unset di service pop3 { ... }

pop3

Determines how the security device evaluates Post Office Protocol version 3 (POP3) traffic. The security device compares actual POP3 traffic with maximum settings of what you consider to be normal POP3 traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **apop_length** *number* Specifies the maximum number of bytes for an Authenticated Post Office Protocol (APOP) command, which a POP3 user issues when authenticating himself to a POP3 mailserver.

  Valid range: 1 - 8192; default: 100.
- **failed_logins** *number* Specifies the maximum number of failed login attempts per minute to a POP3 server from a single host.

  Valid range: 2 - 100; default: 4.
- **line_length** *number* Specifies the maximum number of bytes for any POP3 line.

  Valid range: 1 - 8192; default: 512.
- **max_msg_num** *number* Specifies the maximum number of messages in a single mailbox on a POP3 server.

  Valid range: 100 - 10,000,000; default: 10,000,000.
- **pass_length** *number* Specifies the maximum number of bytes in a POP3 password.

  Valid range: 1 - 8192; default: 64.
- **user_length** *number* Specifies the maximum number of bytes in a POP3 username.

  Valid range: 1 - 8192; default: 64.

radius

get di service radius [...]
set di service radius { ... }
unset di service radius { ... }

radius

- **failed_auth** *number* Specifies the maximum number of failed login attempts per minute to a RADIUS server from a single host.

  Valid range: 2 - 100; default: 8.

smb

get di service smb { ... }
set di service smb { ... }
unset di service smb { ... }

smb — Determines how the security device evaluates Server Message Block (SMB) traffic. The security device compares actual SMB traffic with maximum settings of what you consider to be normal SMB traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **failed_logins** *number* Specifies the maximum number of failed login attempts per minute to an SMB server from a single host.

  Valid range: 2 - 100; default: 8.
- **regkey_length** *number* Specifies the maximum number of bytes in an SMB registry key.

  Valid range: 32 - 64,535; default: 8192.

smtp

get di service smtp { ... }
set di service smtp { ... }
unset di service smtp { ... }

smtp — Uses the Simple Mail Transfer Protocol (SMTP) threshold parameters to control how the security device handles SMTP packets. The threshold parameters define the boundaries of normal SMTP traffic. Traffic that exceeds these boundaries is considered abnormal and might contain protocol anomalies.

- **check_headers_in_body { 0 | 1 }** Enables or disables the inspection of SMTP traffic for email headers in the body of an email message, which can occur when a bounced message contains an attachment.
  A value of 0 disables checking for SMTP headers in the body of an email message, and 1 enables it. By default, this option is disabled.
- **cmdline_length** *number* Specifies the maximum number of bytes in any command line sent from an SMTP client within an SMTP message envelope.

  Valid range: 1 - 8192; default: 1024.
- **content_filename_length** *number* Specifies the maximum number of bytes for the name of a file in a content-disposition filename parameter in an SMTP header. For information about the content-disposition header field, refer to RFC 2183, *Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field*.

  Valid range: 1 - 1024; default: 128.
- **content_name_length** *number* Specifies the maximum number of bytes in the content-type name attribute in an SMTP header. Two examples of content-type names are *text/plain; name="CLI.pdf"* and *application/zip; name="nsremote.zip"*. For information about various content types, see RFC 2046 *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*.

  Valid range: 1 - 1024; default: 128.

- **domain_length** *number* Specifies the maximum number of bytes in the domain-name component of the forward-path field in an RCPT command or reverse-path field in a MAIL command in an SMTP message envelope. The forward-path field indicates the destination mailbox. The reverse-path field indicates the sender's mailbox. The mailbox name consists of two parts: usr_name@domain_name

  Valid range: 1 - 8192; default: 64.

- **multipart_depth** *number* Specifies the number of nested elements in a multipart content type. For an example, refer to "Appendix A – A Complex Multipart Example" in RFC 2049, *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*.

  Valid range: 1 - 16; default: 4.

- **num_rcpt** *number* Specifies the maximum number of recipients for an SMTP message.

  Valid range: 1 - 1000; default: 100.

- **parse_cnt_length** *number* Specifies the maximum number of bytes of encoded MIME data that the security device must decode.

  Valid range: 1 - 8192; default: 128.

- **path_length** *number* Specifies the maximum number of bytes that can appear in the forward-path field in an RCPT command or in the reverse-path field in a MAIL command in an SMTP message envelope. The forward-path typically consists of the destination mailbox. The reverse-path typically consists of the sender's mailbox.

  Valid range: 1 - 8192; default: 256.

- **replyline_length** *number* Specifies the maximum number of bytes in a reply line sent from an SMTP server. The total length includes the three-digit reply code and the <CRLF>.

  Valid range: 1 - 8192; default: 512.

- **textline_length** *number* Specifies the maximum number of bytes in a single SMTP text line, including the <CRLF>.

  Valid range: 1 - 8192; default: 512.

- **user_length** *number* Specifies the maximum number of bytes in a username component of the forward-path field in an RCPT command or in the reverse-path field in a MAIL command in an SMTP message envelope. The forward-path field indicates the destination mailbox. The reverse-path field indicates the sender's mailbox. The mailbox name consists of two parts: usr_name@domain_name

  Valid range: 1 - 8192; default: 256.

syslog

```
get di service syslog [ ... ]
set di service syslog { ... }
unset di service syslog { ... }
```

syslog	**validate_timestamp { 0 | 1 }** Enables (1) or disables (0) the feature that validates RFC 3164, *Compliant Timestamp*, format.

telnet

get di service telnet [...]
set di service telnet { ... }
unset di service telnet { ... }

telnet	**failed_logins** *number* Specifies the maximum number of login failures per minute. Valid range: 2 through 100; default: 4.

tftp

get di service tftp [...]
set di service tftp { ... }
unset di service tftp { ... }

tftp	**filename_length** *number* Specifies the maximum length for the filename. Valid range: 1 through 8192; default: 128.

vnc

get di service vnc [...]
set di service vnc { ... }
unset di service vnc { ... }

vnc

- **failed_logins** *number* Specifies the maximum number of failed logins per minute.

  Valid range: 2 through 100; default; 4.
- **max_cuttext_length** *number* Specifies the maximum cut-text length.

  Valid range: 1 through 65,536; default: 4096.
- **max_name_length** *number* Specifies the maximum length for the display name.

  Valid range: 1 through 1024; default: 128.
- **max_reason_length** *number* Specifies the maximum string length for the reason.

  Valid range: 1 through 2048; default: 512.
- **verify_message { 0 | 1 }** Enables (1) or disables (0) the feature that checks the maximum length of the verify message after the initial handshake.

whois

get di service whois [...]
set di service whois { ... }
unset di service whois { ... }

whois	**request_length** *number* Specifies the maximum length of a request. Valid range: 1 through 1024; default: 128.

ymsg

get di service ymsg { ... }
set di service ymsg { ... }
unset di service ymsg { ... }

ymsg

Determines how the security device evaluates Yahoo! Messenger (YMSG) traffic. The security device compares actual YMSG traffic with maximum settings of what you consider to be normal YMSG traffic. The security device considers any traffic exceeding such settings to be anomalous.

- **max_activity** *number* Specifies the maximum number of bytes in the length of a data-type activity value. Data-type activities include PEERTOPEER, FILEXFER, and TYPING.

  Valid range: 1 - 20; default: 15.

- **max_buddy_list** *number* Specifies the maximum length in bytes of the buddy list that a YMSG server sends.

  Valid range: 20 - 8000; default: 8000.

- **max_challenge** *number* Specifies the maximum length in bytes of the challenge string that a YMSG server sends during the authentication process.

  Valid range: 1 - 1024; default: 84.

- **max_chatroom_msg** *number* Specifies the maximum length in bytes of a message sent in a chat room.

  Valid range: 1 - 8000; default: 2000.

- **max_chatroom_name** *number* Specifies the maximum length in bytes of a YMSG chat-room name.

  Valid range: 1 - 8000; default: 1024.

- **max_conf_msg** *number* Specifies the maximum number of bytes in a YMSG conference-join message.

  Valid range: 1 - 8000; default: 1024.

- **max_conference_name** *number* Specifies the maximum length in bytes of a YMSG conference-session name.

  Valid range: 1 - 8000; default: 1024.

- **max_cookie_length** *number* Specifies the maximum number of bytes in the cookie that a YMSG server sends to a client.

  Valid range: 1 - 1000; default: 400.

- **max_crypt** *number* Specifies the maximum number of bytes in the encrypted password sent during the YMSG authorization process.

  Valid range: 1 - 8000; default: 1024.

- **max_file_name** *number* Specifies the maximum length in bytes of the name of a file that YMSG peers can transfer to each other.

  Valid range: 1 - 8000; default: 1000.

- **max_group_name** *number* Specifies the maximum length in bytes for a name of a group of buddies.

  Valid range: 1 - 1024; default: 84.

- **max_mail_address** *number* Specifies the maximum length in bytes of the address in an email message that a YMSG server sends as part of a new email alert.

  Valid range: 1 - 1024; default: 84.

- **max_mail_subject** *number* Specifies the length in bytes of the subject line in an email message that a YMSG server sends as part of a new email alert.

  Valid range: 1 - 1024; default: 128.

- **max_message_size** *number* Specifies the maximum length in bytes of a YMSG instant message.

  Valid range: 1 - 1024; default: 128.

- **max_url_name** *number* Specifies the maximum length in bytes of a uniform resource locator (URL).

  Valid range: 1 - 8000; default: 1024.

- **max_user_name** *number* Specifies the maximum length in bytes of a YMSG username.

  Valid range: 1 - 1024; default: 84.

- **max_webcam_key** *number* Specifies the maximum number of bytes in the webcam key that YMSG uses to support webcam transmissions.

  Valid range: 1 - 1024; default: 124.

- **max_yahoo_message** *number* Specifies the maximum total length in bytes of a YMSG instant message.

  Valid range: 200 - 8192; default: 8192.

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 consisting of a range of Internet Protocol (IP) addresses defined on a Layer 3 security zone interface, extended interface, or numbered tunnel interface. When multiple security devices are in a High Availability (HA) 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 a 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 a SNMP alarm. The default value for this threshold is 10% 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 **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 IPv4 address in the format 123.123.123.

DNS translation is supported in all the following applications:

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

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

Syntax

clear

```
clear [ cluster ] dns
    [
    ddns [ id id_num ] |
    proxy |
    server-select [ domain dom_name ]
    ]
```

exec

```
exec dns
    {
    ddns [ id id_num ] |
    refresh
    }
```

get

```
get dns
    {
    ddns [ id id_num ] |
    host { cache | report | server-list | settings } |
    name dom_name |
    proxy |
    server-select
    }
```

set

```
set dns
    {
    ddns
      [
      enable |
      id id_num
        }
        [ server name_str ] server-type { ddo | dyndns }
          [ refresh-interval number ]
            [ minimum-update-interval number ]
              [ clear-text ]
        src-interface interface [ host-name name_str ] |
        username name_str password pswd_str [ agent name_str ]
        }
      ] |
    host
      {
      dns1 ip_addr | dns2 ip_addr | dns3 ip_addr
        [ src-interface interface ] |
      name name_str ip_addr |
      schedule time [ interval number ]
      } |
    proxy [ enable ] |
    server-select domain dom_name
      [
        [ outgoing-interface interface ]
        failover |
        primary-server ip_addr
          [ failover |
          secondary-server ip_addr
            [ failover |
            tertiary-server ip_addr
              [ failover ]
            ]
          ]
      ]
    }
```

Keywords and Variables

cluster

clear [cluster] dns

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

ddns

get dns ddns [id *id_num*] […]
set dns ddns enable
set dns ddns [id *id_num*] […]
unset dns ddns

ddns

Initiates or deletes the DDNS (Dynamic DNS) entry in the DDNS Entries table. Each entry represents a module that allocates all resources needed for DDNS. Deleting an entry frees the resources allocated for the module.

Dynamic DNS (DDNS) is a mechanism that allows clients to dynamically update IP addresses for registered domain names. This is useful when an ISP uses PPP, DHCP, or XAuth to dynamically change the IP address for a CPE router (such as a security device) that protects a web server. Thus, any clients from the internet can access the web server using a domain name, even if the IP address of the CPE router previously changed dynamically.

This is made possible by a DDNS server such as dyndns.org or ddo.jp, which contains the dynamically-changed addresses and their associated domain names. The CPE updates these DDNS servers with this information, periodically or in response to IP address changes.

- **enable** Enables the DDNS module.
- **id** *id_num* Identifies a DDNS entry in the DDNS Entries table. If an entry already exists with this ID number, the **set dns ddns id** *id_num* command updates the server information for that entry. If not, the command creates a new entry.
  - **server** *name_str* The FQDN (Fully-Qualified Domain Name) of the DDNS server. The maximum length is 63 characters.
  - **server-type { ddo | dyndns }** The type (DDO or DYNDNS) of DDNS server.
    - **clear-text** Disables HTTPS. The default is to use HTTPS encryption, for both servers.
    - **refresh-interval** *number* The time interval (expressed in hours) between refreshing of the DDNS entry. The default is 168 hours, and the allowable range is 1-8760 hours.
    - **minimum-update-interval** *number* The minimum period (expressed in minutes) between updates. The default is 10 minutes, and the allowable range is 1-1440 minutes.
    - **src-interface** *interface* The interface through which the device communicates with the DDNS server. The optional **host-name** *name_str* parameter identifies a host name for the security device. **Note:** This value is necessary only if the DDNS server is of type DYNDNS, not DDO.

- **username** *name_str* **password** *pswd_str* [**agent** *name_str*] Identifies the username and password for the DDNS account. The maximum length for each of these settings is 63 characters.
  - **agent** *name_str* Specifies the name of the agent. The default value is:
    string1-string2-id_num, where:
    - string1 the company name
    - string2 the software version
    - id_num the serial number
    The maximum length of the total agent string is 63 characters.

host

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

host

- **cache** Displays the DNS cache table.
- **dns1** *ip_addr* Specifies the primary DNS server.
- **dns2** *ip_addr* Specifies the backup DNS server.
  - **src-interface** *interface* Specifies an interface so that DNS requests packets, although initiated from within the system by the DNS module, are treated as if received externally from the source interface you set. When you specify a src-interface, DNS request packets, like all user data packets, trigger firewall policy lookup and are handled according to the rules of the policy. The source interface can be any interface that matches the zone.
- **name** The domain name of the host, listed in the DNS table.

  Using the **name** option with **set** places an entry in the DNS table, representing a host device with a host name and IP address. This allows you to reach the host from the security device using the host name. For example, executing **set dns host name acme 2.2.2.25** creates a DNS table entry for a host at address *2.2.2.25*, with a host name 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.
- **server-list** Displays the IP addresses of hosts currently designated as DNS servers.
- **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 IP address **1.2.2.45**:

set dns host dns1 1.2.2.45

Example 2: The following command schedules a refresh time at **23:59** each day and a DNS table refresh interval of 12 hours:

set dns host schedule 23:59 interval 12

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 FQDN (Fully-Qualified Domain Name) contained in each DNS query. This process is called *proxy DNS*. ■ **primary-server** *ip_addr* ■ **secondary-server** *ip_addr* ■ **tertiary-server** *ip_addr* 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 partially-filled or fully-filled 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. **Note:** Before you can use the server-select options, you must enable DNS proxy using the **set dns proxy** and **set dns proxy enable** commands. For more information on proxy DNS, see proxy on page 163.

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 IP address 2.2.2.2. 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 2.2.2.2
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 security device. This name allows other devices to access the security device through a Domain Name System (DNS) server, thus identifying the device without using an explicit Internet Protocol (IP) address.

Syntax

get

get domain

set

set domain *name_str*

Keywords and Variables

Variable Parameter

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

downgrade

Use the **downgrade** command to downgrade the ScreenOS firmware from ScreenOS 5.0.X to ScreenOS 4.0.X.

To use this command to perform a downgrade, you must have the following.

- Root or Read-Write privileges to the security device
- A console connection to the security device
- A TFTP server application running on your computer
- An Ethernet connection from your computer to the security device (to transfer data from the TFTP server on your computer)
- A ScreenOS 4.0.X image file saved to the TFTP server folder on your computer
- A configuration file that was saved in ScreenOS 4.0.X (configurations saved in ScreenOS 5.0.0 are not supported by ScreenOS 4.0.X)

For information on the downgrade process, refer to the 5.4.0 release notes.

CAUTION: Execute this command with extreme caution. Before execution, refer to the 5.4.0 release notes.

Syntax

exec

exec downgrade

Keywords and Variables

None.

Downgrades and NetScreen-Security Manager

Before downgrading ScreenOS from 5.x to 4.x on a device that uses NetScreen-Security Manager (NSM), execute the following commands:

unset nsmgmt enable
unset nsmgmt init otp
unset nsmgmt init id
unset nsmgmt server primary
del nsmgmt keys
save

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* | max-frame-size=*frame_size* }

Keywords and Variables

Variable Parameter

set envar *string*
unset envar *string*

string	The location of the environment variables files.

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

set envar config=slot2:file2.cfg

max-frame-size

set envar max-frame-size=9830

max-frame-size	Defines the maximum frame size (MTU) that the security device can process; this is a system-wide definition and is not per interface. You must reboot the system for the new setting to take effect. This feature is only available on certain platforms and with certain security modules.. When using this feature, the following statements apply to the new environment: ■ Deep inspection (DI) is not supported. ■ Packets sent through aggregate interfaces might be out of order. ■ If the system is placed in jumbo frame mode, jumbo frame NSRP forwarding is not supported. ■ Maximum firewall or VPN throughput requires at least four sessions (for firewall) or tunnels (for VPN).

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 ■ (max-frame-size) Maximum size of frames

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, Tear Drop, and Ping of Death attacks.
- **Error:** Messages for admin login failures.
- **Information:** Any kind of message not specified in other categories.
- **Notification:** Messages concerning traffic logs and link-status 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_num_high [ -id_num_low ] ]
          [ 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_string ]
          [ end-date date_string ]
      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* [...]
get event sort-by dst-ip [...]

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.

sort-by

get event sort-by { ... }

sort-by	Directs the device to sort event logs by date, sorce IP address, distination IP address, or time.

src-ip

get event src-ip *ip_addr1* [...]
get event sort-by src-ip *ip_addr1* [...]

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 start-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_string*] [end-date *date_string*]
get event sort-by *date* [start-date *date_string*] [end-date *date_string*]

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.

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->

Notes

When issuing the **exit** command at the top command level (that is, 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. The **get failover** command allows you to view the status of the failover settings.

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).

enable

set failover enable
unset failover enable

enable	Enables failover mode on the security device.

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 hold-down interval is 30 seconds. The range is 1-32767 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 interface.

type

set failover type { track-ip | tunnel-if }
set failover type route vrouter *vrouter ip_addr/mask*

type	Specifies the type of event that determines interface failover. You can specify the following types: ■ **route** monitors a known route's status. The route entry can be propagated by a dynamic routing protocol, such as BGP or OSPF. If a BGP adjacency is lost, the security device removes all routes learned from that BGP peer. If the route entry is not active for a period of time that exceeds the hold-down time, the security device triggers an interface failover. This feature requires an exact address match in the specified vrouter and the route must be active to avoid failover. ■ **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 or USB storage device.

Syntax

get

get file [*filename* | info]

Keywords and Variables

Variable Parameters

delete file *dev_name*:/*filename*
get file *filename*

dev_name:/filename	Deletes the file with the name *filename* from the flash card memory (*dev_name* = flash) or the USB storage device (dev_name = usb).
filename	Defines the file name stored in the flash card memory or USB storage device.

Example: 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.

firewall

Use the **firewall** commands to enable or disable logging of dropped packets targeting an interface address on the security device or to specify thresholds for packets sent to the CPU by a Packet Process Unit (PPU).

NOTE: Security devices perform most firewall services at the security-zone level. You configure individual zones to perform these services. For more information, see "zone" on page 695.

Syntax

get

```
get firewall
    [
      ppp-threshold packet-drop { non-ip | other-ip | system-critical }
    ]
```

set

```
set firewall
    {
    log-self [ exclude ] [ icmp | ike | multicast | snmp ] |
    ppu-threshold packet-drop
      { { non-ip | other-ip | system-critical }
        number1 number2 }
    }
```

Keywords and Variables

firewall

get firewall

firewall	Displays the settings for logging dropped ICMP, IKE, multicast, and SNMP packets destined for the security device. Log entries appear in the self log.

log-self

set firewall log-self [exclude] [icmp | ike | multicast | snmp]
unset firewall log-self [exclude] [icmp | ike | multicast | snmp]

log-self

Directs the security device to log or not log dropped packets and pings in the self log. Using the **exclude** switch directs the device not to perform logging at all or for specified traffic types.

- **icmp** Enables or disables logging of ICMP (Internet Control Message Protocol) packets.
- **ike** Enables or disables logging of dropped IKE (Internet Key Exchange) packets.
- **multicast** Enables or disables logging of multicast packets.
- **snmp** Enables or disables logging of dropped Simple Network Management Protocol (SNMP) packets.

Entering the **set firewall log-self** command without any other keywords enables logging to the self log. (By default, logging to the self log is enabled.) Entering the **unset firewall log-self** command without any other keywords disables it.

ppu-threshold

get firewall ppu-threshold packet-drop { ... }
set firewall ppu-threshold packet-drop { ... }
unset firewall ppu-threshold packet-drop { ... }

ppu-threshold

Defines protection thresholds for the Packet Process Units (PPU), which forward packets to the flow CPU. PPU protection thresholds determine how many packets of a particular type the PPU can send to the CPU before the device begins to drop subsequent packets of that type. This feature protects the security device from CPU overload.

It processes three categories of traffic:

- **non-ip** *number1 number2* Packets that do not use IP protocol.
- **other-ip** *number1 number2* IP packets carrying contents other than TCP or UDP.
- **system-critical** *number1 number2* System-critical IP packets, which includes BGP, OSPF, RIP, SNMP, NSM Agent, SNMP, SIP, and H323 traffic.

When the packet arrival rate for a specified category exceeds threshold number1, the device drops subsequent packets randomly. The probability of packet-dropping grows linearly with the subsequent packet arrival rate for that category. When the packet arrival rate exceeds threshold number2, the device drops all subsequent packets that exceed the threshold.

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-check |
tcp-syn-check-in-tunnel
}

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 (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, then 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 query (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 packet flow counter for the interface.

By default, **allow-dns-reply** is disabled. Enabling **allow-dns-reply** directs the security 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-MSS (TCP-Maximum Segment Size) value for all TCP packets for network traffic. This also sets the TCP-MSS for IPSec VPN traffic if the **tcp-mss** option (described below) is not set. If you enter the **set flow tcp-mss** command, that setting overrides the **all-tcp-mss** option for VPN traffic. The TCP-MSS range can be from 0 to 65,535 bytes. By default, the **all-tcp-mss** option is unset.

check tcp-rst-sequence

set flow check tcp-rst-sequence
unset flow check tcp-rst-sequence

check tcp-rst-sequence	Checks that the TCP sequence number in a TCP segment with the RST bit enabled matches the previous sequence number for a packet in that session or is the next higher number incrementally. If the sequence number does not match either of these expected numbers, the security device drops the packet and sends the host a TCP ACK segment with the correct sequence number. By default, this check is disabled.

gre-in-tcp-mss

set flow gre-in-tcp-mss [*number*]
unset flow gre-in-tcp-mss

gre-in-tcp-mss	Enables and specifies the TCP-MSS (TCP-Maximum Segment Size) for Generic Routing Encapsulation (GRE) packets that are about to go into an IPSec VPN tunnel. If the security device receives a GRE-encapsulated TCP packet with the SYN bit and TCP-MSS option set and the TCP-MSS option specified in the packet exceeds the TCP-MSS specified by the security device, then the security device modifies the TCP-MSS value accordingly. By default, a TCP-MSS for GRE packets is not set. When it is enabled, the default TCP-MSS is 1320 bytes. The TCP-MSS can be between 64 and 1420 bytes inclusive.

gre-out-tcp-mss

set flow gre-out-tcp-mss [*number*]
unset flow gre-out-tcp-mss

gre-out-tcp-mss	Enables and specifies the TCP-MSS (TCP-Maximum Segment Size) for Generic Routing Encapsulation (GRE) packets that are leaving an IPSec VPN tunnel. If the security device receives a GRE-encapsulated TCP packet with the SYN bit and TCP-MSS option set and the TCP-MSS option specified in the packet exceeds the TCP-MSS specified by the security device, then the security device modifies the TCP-MSS value accordingly. By default, a TCP-MSS for GRE packets is not set. When it is enabled, the default TCP-MSS is 1320 bytes. The TCP-MSS can be between 64 and 1420 bytes inclusive.

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 forward traffic arriving through a VPN tunnel to a mapped IP (MIP) address on one tunnel interface to the MIP host at the end of another VPN tunnel. The two tunnels form a hub-and-spoke configuration, with the traffic looping back on the same outgoing interface. This option only has an effect when the outgoing 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*) that the security device keeps an initial TCP session in the session table before dropping it, or until the device receives a FIN or RST packet. When *number* is less than or equal to 5, the range of time is in 60-second intervals, from 60 seconds to 300 seconds; otherwise the range of time is in 20-second intervals, from 20 seconds to 300 seconds.

Example: The following command sets the **initial-timeout** value to 300 seconds:

set flow initial-timeout 5

Example: The following command sets the **initial-timeout** value to 280 seconds:

set flow initial-timeout 280

mac-cache

set flow mac-cache mgt
unset flow mac-cache mgt

mac-cache mgt	Caches the source MAC address from incoming administrative traffic for use when replying. This option might be necessary when the security device uses source-based routing. By default, this option is unset.

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. By default, this option is enabled.

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 bytes inclusive. For example, if a received packet is 1500 bytes and **max-frag-pkt-size** is 1460 bytes, the device generates two fragment packets. The first is 1460 bytes and the second is 40 bytes. If you reset **max-frag-pkt-size** to 1024, the first fragment packet is 1024 bytes and the second is 476 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

multicast install-hw-session

set flow multicast install-hw-session
unset flow multicast install-hw-session

multicast	Enables and disables the hardware install multicast session.

no-tcp-seq-check

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

no-tcp-seq-check	When this command is set, the security device does not check sequence numbers in TCP segments during stateful inspection. When unset, TCP sequence number checking is enabled. The security device detects the window scale specified by both source and destination hosts in a session and adjusts a window for an acceptable range of sequence numbers according to their specified parameters. The security device then monitors the sequence numbers in packets sent between these hosts. If the security device detects a sequence number outside this range, it drops the packet. Starting with ScreenOS 5.1.0, the default behavior of security devices is to monitor sequence numbers in TCP segments. However, when upgrading from an earlier ScreenOS release, the security device maintains the existing setting for TCP sequence number checking. Therefore, if it was disabled before upgrading, it remains disabled after upgrading.

path-mtu

set flow path-mtu
unset flow path-mtu

path-mtu	Determines whether the security device sends the source host an ICMP message that a packet size is too large (ICMP type 3, code 4 "Fragmentation needed and DF set") when it receives a packet meeting the following conditions: ■ The Don't Fragment (DF) bit is set in the IP header. ■ The packet is intended for IPSec encapsulation. ■ The size of the packet after encapsulation exceeds the maximum transfer unit (MTU) of the egress interface, which is 1500 bytes. When you enable (set) the path-mtu option, the security device sends the source host the above ICMP message. When you disable (unset) this option, the security device ignores the DF bit, encapsulates the packet, fragments the packet so that none of the fragmented packets exceeds the MTU of the egress interface, and forwards them through the appropriate VPN tunnel. By default, this option is disabled.

perf

get flow perf

perf	Displays performance information.

route-change-timeout

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

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

syn-proxy syn-cookie	Sets the flow from 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

get flow tcp-mss
set flow tcp-mss [*number*]
unset flow tcp-mss

tcp-mss	Sets the TCP-MSS (TCP-Maximum Segment Size) value for all TCP SYN packets for IPSec VPN traffic. The security device modifies the MSS value in the TCP packet to avoid fragmentation caused by the IPSec operation.

tcp-rst-invalid-session

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

tcp-rst-invalid-session	Marks a session for immediate termination when it receives a TCP reset (RST) segment. By default, this command is unset. When unset, the security device applies the normal session timeout interval—for TCP, session timeout is 30 minutes; for HTTP, it is 5 minutes; and for UDP, it is 1 minute.

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. By default, the security device checks that the SYN bit is set in the first packet of a session. If it is not set, the security device drops it.

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.

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* [add *name_str*] [comment *string*] [hidden]
[ipv6 [add *name_str*] [comment *string*] [hidden]] |
service *grp_name* [add *name_str*] [comment *string*] [hidden]
}

Keywords and Variables

add

set group address zone *grp_name* [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*.

Example 1: 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

Example 2: 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 on zones, see "Zone Names" on page B-I.

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 the 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"

hidden

set group address *zone grp_name* [hidden]
set group service *grp_name* [hidden]

hidden	Specifies that the service group or address group is a hidden service or group. We strongly recommend that you do not hide service groups or address groups.

ipv6

set group address *zone grp_name* [...] [ipv6] [...]

ipv6	Specifies that the address group is an IPv6 group.

Example: The following command creates an address group named *engineering* for the Trust zone, and specifies that it is a hidden group:

set group address trust engineering ipv6

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

Variable Parameters

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

gtp

Use the **gtp** commands to delete existing GTP tunnels on the security device, remove GTP inspection-object configurations, obtain configuration information, or configure a GTP object.

Syntax

clear

clear gtp tunnel { *number* | all }

get

get gtp { configuration [*name_str*] | tunnels }

get (within an object context)

get configuration

set

set gtp configuration *name_str*

set (within an object context)

```
set gtp configuration name_str
    set
    {
    apn { string { drop | pass | select [ ms | net | vrf ] } } |
    drop
      {
      create-pdp |
      crt-aa-pdp |
      data-record |
      del-aa-pdp |
      delete-pdp |
      echo |
      error-indication |
      failure-report |
      fwd-relocation |
      fwd-srns-context |
      g-pdu |
      identification |
      node-alive |
      note-ms-present |
      pdu-notification |
      ran-info |
      redirection |
      relocation-cancel |
```

```
    send-route |
    sgsn-context |
    supported-extension |
    update-pdp |
    ver-not-supported
    }
    [ number ]
    } |
gtp-in-gtp-denied |
imei-sv string
    {
    apn string { drop | pass | select { ms | net | vrf } } |
    mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
    } |
limit { rate number | tunnel number } |
log
    {
    forwarded { basic [ number ] | extended [ number ] } |
    prohibited { basic [ number | extended [ number ] } |
    rate-limited { basic [ number | extended [ number ] } |
    state-invalid { basic [ number ] | extended [ number ] }
    |traffic-counters [ byte-counts ] |
    tunnel-limited { [ number ] | extended [ number ] }
    } |
max-message-length number |
min-message-length number |
notify ip_addr
    {
    [ port port_num ]
    src-interface interface context id_num [ md5-authentication password ]
    } |
rai string
    {
    apn string { drop | pass | select { ms | net | vrf } } |
    imei-sv string
        {
        apn string { drop | pass | select { ms | net | vrf } } |
        mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
        } |
    mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } } |
    uli string
        {
        apn string { drop | pass | select { ms | net | vrf } } |
        imei-sv string
            {
            apn string { drop | pass | select { ms | net | vrf } } |
            mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
            }
        } |
    }
rat string
    {
    apn string { drop | pass | select { ms | net | vrf } } |
    imei-sv string
        {
        apn string { drop | pass | select { ms | net | vrf } } |
        mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
```

```
      } |
    mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } } |
    rai string
      {
      apn string { drop | pass | select { ms | net | vrf } } |
      imei-sv string { apn string { drop | pass | select { ms | net | vrf } } |
      mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } } |
      uli string
        {
        apn string { drop | pass | select { ms | net | vrf } } |
        imei-sv string
          {
          apn string { drop | pass | select { ms | net | vrf } } |
          mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
          }
        mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
        } |
    uli string
      {
      apn string { drop | pass | select { ms | net | vrf } } |
      imei-sv string
        {
        apn string { drop | pass | select { ms | net | vrf } } |
        mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } } |
        }
      mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
      }
    } |
  remove-r6 |
  seq-number-validated |
  teid-di |
  timeout number |
  trace
    {
    imsi number |
    max-active number [ save-length number ] |
    msisdn number
    }
  uli string
    {
    apn string { drop | pass | select { ms | net | vrf } } |
    imei-sv string
      {
      apn string { drop | pass | select { ms | net | vrf } } |
      mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } } |
      }
    mcc-mnc string { apn string { drop | pass | select { ms | net | vrf } }
    }
  }
```

Keywords and Variables

apn

set apn *string* { drop | pass | selection }
unset apn *string*

apn

The **set** and **unset** commands allow access or deny access to specific Access Point Names (APNs).

- *string* Sets an APN suffix such as "netscreen.com.mcc123.mnc456.gprs".
- **drop** Specifies to deny GTP packets from all Selection Modes for this APN.
- **pass** Specifies to permit GTP packets from all Selection Modes for this APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

Note: Because APN filtering is based on a perfect match, using the wildcard * when setting an APN suffix can prevent the inadvertent exclusion of APNs you would otherwise authorize. The security device automatically permits all other APNs that do not match.

configuration

get gtp configuration

configuration

Displays information on the configuration of the current GTP Inspection.

drop

set drop *message_type* [version *number*]
unset drop *message_type* [version *number*]

drop — Displays information on the configuration of the current GTP Inspection.

- *number* Specifies the GTP release version number for the specified message type. The possible versions are **0** (for GTP 97) or **1** (GTP 99). If you do not set a version number, the device drops all packets of the specified message type for both GTP release versions.

The following lists CLI keywords that each represent a GTP message type. A GTP message type includes one or many messages. When you set or unset a message type, you automatically permit or deny access to all messages of the specified type.

- **create-pdp** Represents Create PDP Context Request and Create PDP Context Response messages.
- **crt-aa-pdp** Represents Create AA PDP Context Request and Create AA PDP Context Response messages.
- **del-aa-pdp** Represents Delete AA PDP Context Request and Delete AA PDP Context Response messages.
- **delete-pdp** Represents Delete PDP Context Request and Delete PDP Context Response messages.
- **echo** Represents Echo Request and Echo Response messages.
- **error-indication** Represents Error Indication messages.
- **failure-report** Represents Failure Report Request and Failure Report Response messages.
- **fwd-relocation** Represents Forward Relocation Request, Forward Relocation Response, Forward Relocation Complete, and Forward Relocation Complete Acknowledge messages.
- **fwd-srns-context** Represents Forward SRNS Context Request and Forward SRNS Context Response messages.
- **g-pdu** Represents G-PDU and T-PDU messages.
- **identification** Represents Identification Request and Identification Response messages.
- **node-alive** Represents Node Alive Request and Node Alive Response messages.
- **note-ms-present** Represents Note MS GPRS Present Request and Note MS GPRS Present Response messages.

gtp-in-gtp-denied

set gtp-in-gtp-denied
unset gtp-in-gtp-denied

gtp-in-gtp-denied — Enables the security device to detect and drop GTP packets that contain another GTP packet in its message body.

imei-sv

set imei-sv *string* apn *string* { ... }
un set imei-sv *string* apn *string* { ... }

imei-sv	Enables the security device to detect and drop GTP packets that contain International Mobile Equipment Identity-Software Version (IMEI-SV) information element.

- *number* Specifies an IMEI-SV name.
- *string* Specifies an APN.
- **pass** Enables the security device to permit GTP packets from all Selection Modes for the specified APN.
- **drop** Enables the security device to deny GTP packets from all Selection Modes for the specified APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

limit

set limit { rate *number* | tunnel *number* }
unset limit { rate | tunnel }

limit	The **set** or **unset** command configures or removes the following types of limits:

- **rate** *number* Specifies a limit in packets per second for GTP-C messages.
- **tunnel** *number* Specifies a limit in the number of GTP tunnels that can be created in the current GTP inspection object per GSN.

log

set log { ... }
unset log { ... }

log	Instructs the security device to log or cease logging the following information: ■ **forwarded** A packet that the security device transmitted because it was valid. ■ **prohibited** A packet that the security device dropped because it was invalid. ■ **rate-limited** A packet that the security device dropped because it exceeded the maximum rate limit of the destination GSN. ■ **state-invalid** A packet that the security device dropped because it failed stateful inspection. ■ **traffic-counters** The number of user data and control messages the security device received from and forwarded to the GGSNs and SGSNs it protects. ■ **byte-counts** The number of bytes the security device received from and forwarded to the GGSNs and SGSNs it protects instead of the number of messages. ■ **tunnel-limited** A packet that the security device dropped because the maximum limit of tunnels for the destination GSN was reached, thus a tunnel could not be established. The following options apply to all the **set log** commands listed above except **traffic-counters**: ■ **basic** Specifies to log the basic Information Elements (IEs) of the GTP message. ■ **extended** Specifies to log other IEs in addition to the basic IEs of the GTP message.

max-message-length

set max-message-length *number*
unset max-message-length

max-message-length	Sets the maximum message payload length (in bytes) the security device accepts for a GTP message. The default maximum message length is 65535 bytes.

mcc-mnc

set mcc-mnc *string* apn *string* { ... }
unset mcc-mnc *string* apn *string*

mcc-mnc	By default, the security device grants access to any International Mobile Station Identity (IMSI) prefix. An IMSI prefix consists of a Mobile Country Code (MCC) and a Mobile Network Code (MNC). The **set** and **unset** commands allow or deny specific IMSI prefixes. These commands only apply to create pdp context request GTP messages. The MCC-MNC pair can be five or six digits. You can filter GTP packets based on the combination of an IMSI prefix and an APN.

- *number* Specifies an IMSI prefix.
- *string* Specifies an APN.
- **pass** Enables the security device to permit GTP packets from all Selection Modes for the specified APN.
- **drop** Enables the security device to deny GTP packets from all Selection Modes for the specified APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

min-message-length

set min-message-length *number*
unset min-message-length

min-message-length	Sets the minimum message payload length (in bytes) the security device accepts for a GTP message. The default minimum message length is 0 bytes.

notify

set notify *ip_addr* { ... }
unset notify

notify

The set command enables the GTP firewall (the client) to notify the Gi firewall (the server) of the overbilling attack. Such notification directs the server to drop the unwanted traffic. The unset command disables the notification feature on the GTP firewall.

- *ip_addr* The IP address of the Gi firewall (server).
- **port** *port_num* The port number on which the Gi firewall receives notification messages.
- **src-interface** *interface* The interface from which the GTP firewall sends Overbilling Attack notification to the Gi firewall.
- **context** *id_num* The number that identifies the context. Note that the same context must exist on the Gi firewall.
- **md5-authentication** *password* The MD5 authentication password.

rai

set rai *string* apn *string* { … }
unset rai *string* apn *string* { … }

rai	Enables the security device to detect and drop GTP packets that contain the RAI Information Element.

- *number* Specifies an RAI value.
- *string* Specifies an APN.
- **pass** Enables the security device to permit GTP packets from all Selection Modes for the specified APN.
- **drop** Enables the security device to deny GTP packets from all Selection Modes for the specified APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

rat

set rat *string* apn *string* { … }
unset rat *string* apn *string* { … }

rat	Enables the security device to detect and drop GTP packets that contain the RAT Information Element.

- *number* Specifies an RAT value.
- *string* Specifies an APN.
- **pass** Enables the security device to permit GTP packets from all Selection Modes for the specified APN.
- **drop** Enables the security device to deny GTP packets from all Selection Modes for the specified APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

remove-r6

set remove-r6
unset remove-r6

remove-r6	Enables the security device to detect and remove 3GPP-specific attributes from the GTP packet header when the packet passes into a 2GPP network. This allows you to retain interoperability in roaming between 2GPP and 3GPP networks.

seq-number-validated

set seq-number-validated
unset seq-number-validated

seq-number-validated	Enables or disables the GTP Sequence Number Validation feature.

teid-di

set teid-di *number*
unset teid-di *number*

teid-di	Enables the security device to perform deep inspection on the tunnel endpoint ID (TEID) in G-PDU data messages.

timeout

set timeout *number*
unset timeout

timeout	Sets the tunnel timeout value in hours. The default is 24 hours. Via the process of stateful inspection, if a security device detects no activity in a tunnel for a specified period of time (timeout), it removes the tunnel from the state table.

trace

set trace { ... }
unset trace { ... }

trace — Enables the security device to identify and log the contents of GTP-U or GTP-C messages based on IMSI prefixes or Mobile Station-Integrated Services Data Network (MS-ISDN) identification.

- **imsi** *number* Indicates the IMSI prefix for which you want the security device to trace GTP packets.
- **max-active** *number* Specifies the maximum number of subscribers that the security device can trace concurrently for the current GTP inspection object. The default value is 3 and the range is 1 to 20.
  - **save-length** *number* Specifies the number of bytes of data to log for GTP packets containing user data. You can log partial or complete packets. The default value is 0, which means that the security device does not log any of the content from a GTP-U packet.
- **msisdn** *number* Indicates the MS-ISDN for which you want the security device to trace GTP packets.

tunnel

clear gtp tunnel { *number* | all }
get gtp tunnel

tunnel — The get command displays information on active tunnels on the security device.

The clear command deletes tunnels, thus terminating the connection between the communicating parties. The following specifies which tunnels are deleted:

- *number* Tunnel index (or tunnel ID number)—specifies which tunnel to delete. The security device assigns an index to each tunnel and uses this number internally.
- **all** Specifies to delete all tunnels on the security device.

uli

set uli *string* apn *string* { ... }
unset uli *string* apn *string* { ... }

uli — Enables the security device to screen subscriber's requested content, before allowing a content download, based on the User Location Information (ULI) IE.

- *number* Specifies an ULI value.
- *string* Specifies an APN.
- **pass** Enables the security device to permit GTP packets from all Selection Modes for the specified APN.
- **drop** Enables the security device to deny GTP packets from all Selection Modes for the specified APN.
- **selection** Specifies one of the following Selection Modes for the APN:
  - **ms** The APN is provided by a mobile station (MS) and the user-subscription is not verified.
  - **net** The APN is provided by a network and the user-subscription is not verified.
  - **vrf** The APN is provided by a network or an MS and the user-subscription is verified.

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 such as ns500gate and a domain name such as juniper (see “domain” on page 165), you can use the hostname and domain name (ns500gate.juniper) as a gateway for a VPN tunnel.

Syntax

get

get hostname

set

set hostname *string*

Keywords and Variables

Variable Parameters

string	Sets the name of the security device.

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

set hostname acme

icap

Use the **icap** command to configure your security device to support an external antivirus (AV) scan engine. Your security device communicates with the external AV scan engine using the Internet Content Adaptation Protocol (ICAP).

NOTE: The **set icap** commands are supported at the root level only. The **exec** and **get** commands, however, are supported at both the root and vsys levels.

External AV scanning is supported for HTTP and SMTP. To configure your device to support external AV, in addition to the **icap** commands in this section, you must configure global AV commands and profiles. For more information, see "av" on page 71.

Syntax

exec

exec icap server *name_str* probe |

get

```
get icap
  {
  server [ name_str ] |
  server-group [ name_str ]
  }
```

set

```
set icap
  {
  server name_str
    {
    enable |
    host { ip_addr | name_str } [port number] |
    max-connections number |
    probe-interval number |
    probe-url url_str |
    } |
  server-group name_str [ server name_str ] |
  }
```

Keywords and Variables

Variable Parameters

name_str	Specifies an ICAP server or a group of ICAP servers.

server

exec icap server *name_str* probe
get icap server
get icap server *name_str*
set icap server *name_str* { . . . }
unset icap server *name_str* { . . . }

server — Displays, sets, or performs actions on an ICAP scan-engine server for external AV scanning.

- **enable** Enables the configured ICAP server.
- **host** *IP address* Specifies the IP address or host name of an ICAP server. The maximum string length of an ICAP AV host name is 255 characters.
  - port *number* You may configure a different port from the default 1344 port. The valid range of port numbers is 1024 to 65535.
- **max-connections** *number* Configures the maximum connections to the ICAP server. The upper limit and default values are platform-dependent.
- **probe** Verifies the health of the ICAP server. The device performs a Layer 7 protocol request to verify if the ICAP server is up and displays the result at the console.
- **probe-interval** *number* Configures the ICAP server probe interval in multiples of five seconds. The range of the interval is 0 to 3000 seconds. The default is 10 seconds; zero (0) indicates that the command is disabled.
- **probe-url** *url_str* Configures a URL string to probe the ICAP server. The maximum string length of an ICAP AV probe URL string is 255 characters.

Example: The following command configures an ICAP server, sales_svr, with host IP address 1.1.1.1 and default port 1344. The same ICAP server is configured with a probe interval of 20 seconds and av scan url to /scan. The maximum number of connections to the ICAP server is set to 128:

set icap server sales_svr host 1.1.1.1
set icap server sales_svr probe-interval 20
set icap server sales_svr probe-url /scan
set icap server sales_svr max-connections 128

server-group

get icap server-group
get icap server-group *name_str*
set icap server-group *name_str* server *name_str*
unset icap server-group *name_str* server *name_str*
unset icap server-group *name_str*

server-group	Displays or sets ICAP server group information. Configures an ICAP server group and adds or removes servers from the group. You may also add an ICAP server group to an AV profile.

Example 1: The following commands configure an ICAP server group named juniper-gp and adds ICAP servers (sales_svr, mktg_svr, and eng_svr) to the server group:

set icap server-group juniper-gp server sales-svr
set icap server-group juniper-gp server mktg-svr
set icap server-group juniper-gp server eng-svr

Example 2: The following command removes the ICAP server, eng-svr, from the ICAP server group, juniper-gp:

unset icap server-group juniper-gp server eng-svr

idp

Use the **idp** commands to configure your security device with at least one security module installed for Intrusion Detection and Prevention (IDP). IDP enables your device to detect attacks and prevent attackers from gaining access to your network.

The **idp** commands are issued within a policy context. Use the **idp** commands to enable and disable idp for that policy and to change the IDP mode to active or passive for that policy.

Syntax

set

device(policy:number)-> **set idp [mode tap]**

unset

device(policy:number)-> **unset idp [mode]**

Keywords and Variables

idp

device(policy:number)-> **set idp**
device(policy:number)-> **unset idp**

idp	Enables or disables IDP for the traffic to which the policy applies. By default, IDP is disabled for policies

Example: The following commands create a policy, enter the context of that policy, and then apply IDP to it:

device-> **set policy id 1 from trust to untrust any any any permit**
device-> **set policy id 1**
device(policy:1)-> **set idp**

Example: The following commands enter the context of a previously defined policy, and then disable IDP for it:

device-> **set policy id 1**
device(policy:1)-> **unset idp**

mode

device(policy:number)-> **set idp mode tap**
device(policy:number)-> **unset idp mode**

mode tap	Sets or unsets tap (passive) mode. By default, IDP is in active mode. In active mode, the security device forwards packets to a security module for inspection. If the security device does not detect an attack, it forwards the packet to its destination. If it does detect an attack, the security device performs an IDP action, such as drop, close-server, close-client, and so on. In **tap** mode, the security device copies packets, forwarding the original packet to its destination and forwarding the copy to a security module for inspection. If the security device detects an attack, it makes an event log entry but does not perform any IDP action.

Example: The following commands create a policy, enter the context of that policy, and then apply IDP in tap mode to it:

device-> **set policy id 2 from trust to untrust any any any permit**
device-> **set policy id 2**
device(policy:2)-> **set idp mode tap**

igmp

Use the **igmp** commands to send Internet Group Management Protocol (IGMP) messages, display IGMP settings, monitor IGMP states on a security device, and clear IGMP information.

Syntax

exec

exec igmp interface *interface*
{
query [*mcst_addr* [s_bit] [*ip_addr*]] |
report *mcst_addr* |
leave *mcst_addr*
}

get

get igmp
{
config
group [*ip_addr* [source]] [all] |
interface [all] |
source *ip_addr* |
statistic [all]
}

clear

clear igmp interface *interface* { statistic | group *mcast_addr* | all }

Keywords and Variables

config

get igmp config

config	Displays the configuration settings for IGMP.

group

get igmp group [*mcast_addr* | all]

group	Displays information for the multicast group specified. Specify **all** to display information for all multicast groups.

interface

exec igmp interface *interface* { . . . }
get igmp interface [all]
clear igmp interface *interface* statistic
clear igmp interface *interface* group *mcast_addr* | all

interface	Displays and clears statistics or multicast groups. You can also send IGMP messages for the specified interface.

leave

exec igmp interface *interface* leave *mcst_addr*

leave	Sends a leave message for the specified multicast group. You can execute this command if the interface is in host mode only.

query

exec igmp interface *interface* query [*mcst_addr* [s_bit] [*ip_addr*]]

query	Sends an IGMP query message. If you specify a multicast group address, the interface sends a group-specific query to the specified multicast group. If you do not specify a multicast group address, then the interface sends a general query to the "all hosts" group (224.0.0.1). For IGMPv3, you can specify the following: ■ **s_bit:** Specify this keyword to indicate to other multicast routers that they are to suppress the normal timer updates they perform when they hear a query. ■ *ip_addr:* You can specify a source address. Enter this command only if the interface is in router mode.

Example: The following command sends a general query to the "all hosts" group from interface *ethernet4*:

exec igmp interface ethernet4 query

report

exec igmp interface *interface* report *mcst_addr*

report	Sends an IGMP membership report to the specified group. Enter this command if the interface is in host mode.

Example: The following command sends a membership report to the specified multicast group:

exec igmp interface ethernet4 report 224.2.1.1

source

get igmp source *ip_addr*

source	Displays an IGMP source address.

statistic

get igmp statistic [all]
clear igmp interface *interface* statistic

statistic	Displays or clears IGMP statistics. Enter this command if the interface is in router mode.

ike

Use the **ike** commands to define the Phase 1 and Phase 2 proposals and the gateway for an AutoKey Internet Key Exchange IKE) VPN tunnel, as well as 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 |
ikeid-enumeration [table [detail *src_ip*]]
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 ]
            }
```

IKE Heartbeat

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

Certificates

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

NAT-Traversal

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

XAuth

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

Other IKE Command Switches

set ike
    {
    accept-all-proposal |
    id-mode { ip | subnet } |
    ikeid-enumeration [*threshold_number* [*interval_number*]]
        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 *name_str* address { *ip_addr* | *name_str* } { ... }

address	Defines the remote IKE gateway address either as an IP address, or as a hostname, or 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 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 ns1, define the preshared key as 7a850wq, and specify the Phase 1 security level as compatible:

set ike gateway ns1 address www.juniper.net preshare 7a850wq sec-level compatible

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. 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.

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-ca-hash** *name_str* Specifies the certificate authority (CA) DN hash.
- **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-ca-hash** *name_str* Specifies the certificale authority (CA) distinguished name (DN) to be sent to the IKE peer in the certificate request (CERT REQ) payload. It can be followed by one of the following;
  - SHA-hash of a CA DN—used in place of the actual name of a DN, which can exceed the CLI length limit.
  - all—a CERT REQ payload is sent to the IKE peer for each CA in the trust store.
- **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 { always-send | interval | retry }

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 [*name_str*]
set ike gateway *name_str* { ... } [...]
unset ike gateway *name_str* { ... }

gateway	Configures or displays settings for a remote tunnel gateway.

heartbeat

get ike heartbeat
set ike gateway *name_str* heartbeat { ... }
unset ike gateway *name_str* 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.)

ikeid-enumerator

get ike ikeid-enumeration [table [detail *src_ip*]]
set ike ikeid-enumeration [*threshold_number* [*interval_number*]]
unset ike ikeid-enumeration

ikeid-enumeration — Enables, disables, or displays anti-IKE ID enumeration information for IKE aggressive mode.

- *threshold_number* Specifies the number of attack packets (first messages with an unknown IKE ID) in the specified interval before IKE starts to block the first IKE messages from this IP address. The range is 1 to 65535; the default is 30 packets.
- *interval_number* Specifies the period of time during which the first messages of IKE aggressive mode are blocked after an attack is detected. When the interval expires, the counter is reset and counting restarts. Interval is 10 to 65535 seconds; the default is 10 seconds.
- **table** Displays the number of first messages with unknown IKE IDs.
- **detail** Lists source IP address and interface name of blocked first messages with unknown IKE IDs.

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 security 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 in the local certificate, specify the following for **local-id** (including square brackets): [DinstinguishedName] 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 227).

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.

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.

Example 1: The following command enables NAT traversal for a gateway named *mktg*:

set ike gateway mktg nat-traversal

Example 2: 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 command specifies ethernet3 as the outgoing interface for an IKE gateway named Paris_Gateway at IP address 2.2.2.2. (Authentication uses a preshared key based on the word “scramble”, and the Phase 1 proposals are those for the “compatible” security level for Phase 1 negotiations.)

set ike gateway Paris_Gateway ip 2.2.2.2 outgoing-interface ethernet3 preshare scramble sec-level compatible

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	Specifies or 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.

  Note: When configuring ESP, it is not advisable to set the null switch. Such a configuration may leave IPSec vulnerable to attack.
- 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 security 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 240.

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 Web server 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 1.1.110.200/32
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 1.1.110.200/24
set address trust "inside any" 2.2.2.2/24
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 240.

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 juniper.net 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 *name_str* xauth [...]

xauth

Enables XAuth authentication for the specified IKE gateway configuration.

- The **bypass-auth** option 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.
- The **client** option 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.

  The **username** setting specifies the username for the XAuth client to use on the XAuth server. The **password** setting specifies the password for the XAuth client to use on the XAuth server.
- The **do-edipi-auth** option enables RADIUS authentication based on EDIPI (Electronic Data Interexchange Personal Identifier). With this form of authentication, a user inserts a CAC (Common Access Card) that contains a PKI certificate. Each PKI certificate has an EDIPI ID, which identifies the user.
- The **server** option 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.

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

Defaults

Main mode is the default method for Phase1 negotiations.

The default time intervals before the device 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.

Setting Up a Policy-Based VPN Tunnel

To create a policy-based VPN tunnel for a remote gateway with a static IP address:

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

To set up a VPN tunnel for a dialup user with IKE:

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** commands to remove Internet Key Exchange (IKE)-related cookies from the security device.

Syntax

clear

clear [cluster] ike-cookie { all | *ip_addr* }

Keywords and Variables

Variable Parameter

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*).

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.

infranet

Use the **infranet** commands to set up a security device (Infranet Enforcer) to work with an Infranet Controller in a Unified Access Control (UAC) deployment.

For more information about deploying UAC, refer to the *Unified Access Control Administration Guide.*

Syntax

exec

exec infranet controller { connect | disconnect | IP *ip_addr* keepalive }

NOTE: If you run an **exec infranet controller disconnect** command, the Infranet Enforcer does not attempt to automatically connect with the Infranet Controller. To reconnect, you must run an **exec infranet controller connect** command or restart the Infranet Enforcer.

get

get infranet { controller [name *string*] | enforcer }

set

```
set infranet
    {
    controller
      {
      contact-interval number |
      name string
        [
        ca-idx number |
        cert-subj string |
        host-name string [ port number ] |
        password string |
        src-interface interface |
        timeout number
        url string
        ] |
      timeout action { close | no-change | open } |
      } |
    enforcer mode test |
    policy command string
    }
```

Keyword and Variables

interface	Specifies the name of the interface.
number	Defines the port number or number of seconds for a particular argument.
string	Specifies the name of the Infranet Enforcer or a policy command.

policy

set infranet policy command *string*

command *string*	The policy command pushes the access policies from the Infranet Controller to the security device (Infranet Enforcer). *string* Specifies the policy name.

Example Use the dynamic command designator (**-n**) command to view the access policies in the Infranet Enforcer. For example:

set -n infranet policy command "get all"
Infranet policy command: received, calling jps_exec:
get all
id=1 192.168.2.0/24:* * allow
id=2 10.25.25.2:*;10.25.25.5:* * allow

controller

get infranet controller name *string*
set infranet controller name *string* [...]
unset infranet controller name *string* [...]

connect	Reestablishes a connection with the Infranet Controller.
disconnect	Removes the connection to the currently connected Infranet Controller.
contact-interval	Specifies how often the Infranet Enforcer is going to ping the Infranet Controller for connectivity. The default value is 10 seconds; the range is 3-300 seconds.
IP *ip_addr*	Specifies the IP address of the Infranet Controller. ■ The **keepalive** command is issued periodically by the Infranet Controller to the Infranet Enforcer. If the Infranet Enforcer does not receive a **keepalive** command within a timeout period, the Infranet Enforcer considers the connection to be down.

name *string* — Specifies the name of the Infranet Controller and must be fewer than 32 characters in length.

- **ca-idx** *number* is the number for the Certificate Authority (CA) certificate index.
- **cert-subj** *string* is the string subject that matches the certificate.
- **host-name** *string* [port number] is the host name or IP address of the Infranet Controller. The port number must be **11122**.
- **password** *string* is the NetScreen Address Change Notification (NACN) password of the Infranet Controller.
- **src-interface** *interface* identifies the outgoing interface.
- **timeout** *number* defines the timeout limit for idle Infranet Controller links. The default timeout is 60 seconds; the range is 1-10,000 seconds.
- **url** *string* is the redirect URL (1-512 characters) to which you want the security policy to redirect HTTP traffic. If you do not specify a URL, the security device defaults to the currently-connected Infranet Controller (the default redirect URL is not displyed).

  Use the following format for the URL within double quotes:

  "http://*IP or domain name***/***url path***/?target=%dest-url%"**

  If you specified a **url** *string,* configure a redirect infranet-auth policy (see "policy" on page 455). The security device redirects HTTP traffic to an external webserver instead of to the Infranet Controller. For more information about using the URL string to redirect HTTP traffic, refer to the *Unified Access Control Administration Guide*.

timeout action — Specifies what action to take when the Infranet Controller times out:

- **open** allows existing and new session traffic as allowed by infranet policies.
- **no-change** preserves existing connections and dynamic configuration such as tunnels, but new sessions require authentication.
- **close** removes existing sessions and dynamic configuration and blocks further traffic.

Example: The following command displays information on the Infranet Controller:

get infranet controller name juniper-ic
Name: juniper-ic
Host: 10.150.43.126
Connected to Infranet Controller 0 times
Infranet Controller Connection State:
SSL: Closed
SSH: Closed
(No Keepalives received from Infranet Controller via SSH)
(SSH V2 is active, enabled, and not ready for connections)
Port: 11122
Interface:
Timeout: 60 seconds
Full Subject Name of IC Cert:
CA Hash:
Selected CA:
Redirect URL:

enforcer

get infranet enforcer
set infranet enforcer mode test
unset infranet enforcer mode test

mode test	Places the Infranet Enforcer in **test** mode, where traffic is always allowed and policies are not enforced. However, the permit or deny decision associated with the infranet-auth policies is logged. The **unset** command turns off the test mode and places the Infranet Enforcer in regular mode. In this default mode, the infranet-auth policies are applied and logged based on the auth table entries.

Example: The following command displays information on the Infranet Enforcer:

get infranet enforcer
Mode: Regular

In this mode the infranet-auth policies are enforced and logged based on the auth table entries.

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 “Interface Names” on page A-I.

Syntax

clear

```
clear interface interface
    {
    dot1x statistics |
    extensive |
    frame-relay stats |
    mlfr-uni-nni stats
    }
```

exec

```
exec
    {
    backup interface interface { failover | revert }
    } |
    interface
    {
    ext-loop-back-test [ interval number | round number [ interval l number ] ] |
    all | interface
      {
      phy setting force-sync |
      bert-test [ start | stop ]
      }
    }
```

get

get interface
{
all |
interface
{
association [*mac_addr*] |
bri-options |
clocking |
dhcp
{
client |
relay |
server
{
ip { allocate | idle } |
option
}
} |
dip |
dot1x [statistics] |
e1-options |
extensive |
hold-time
isdn [q921 { statistics | status } q931 { statistics | status }] |
monitor track-ip [ip] |
mip |
ppp |
protocol
{
ospf |
rip [neighbor *ip_addr*] |
igmp
[
config |
group [*ip_addr* [source] [all]] |
source |
statistic [all] |
]
pim [statistics]
}
frame-relay { lmi | pvc | statistics } |
mlfr-uni-nni { config | members | statistics } |
screen |
secondary [*ip_addr*] |
serial-options |
t1-options |
t3-options |
track-ip [ip]
}
}

set (Layer 3 Interfaces)

```
set interface interface
    {
    backup
      {
      activation-delay number |
      auto |
      deactivation-delay number |
      interface interface type { route vrouter string ip_add/mask | track-ip | tunnel-if }
      } |
    bandwidth { egress mbw number | ingress mbw number } |
    description string |
    dhcp server
      {
      auto |
      config
        { next-server-ip [ ip ip_add | option66 ] | updatable [ src-interface ] } |
      disable |
      enable |
      ip ip_add [ mac mac_add | to ip_add ] |
      option
        {
        custom number { integer number | ip ip_add | string string } |
        dns1 | dns2 | dns3 | gateway | netmask | news | nis1 | nis2 | pop3 | smtp |
        wins1 | wins2 { ip_add } |
        domainname | nistag { string } |
        lease number
        } |
      service
      }
    [ ext ip ip_addr/mask ] dip
      {
      interface-ip incoming |
      id_num
        {
        ip_addr1 [ ip_addr2 ] |
        shift-from ip_addr3 [ to ip_addr4 [ ip_addr5 ] ]
        }
          [ fix-port | incoming ]
      }
    gateway ip_addr [ no-default-route ] |
    group |
    ip { ip_addr/mask | manageable } |
    manage
      { ident-reset | nsmgmt | mtrace | ping | snmp | ssh | ssl | telnet | web } |
    manage-ip ip_addr |
    mip ip_addr host ip_addr [ netmask mask ] [ vrouter name_str ] |
    modem
      {
      idle-time number |
      interval number |
      isp string
        {
        account login string password string |
        primary-number number_string [ alternative-number number_string ] |
        priority number
```

```
      } |
   isp-failover
      {
      holddown number |
      type route vrouter vrouter_string ip_add/mask
      } |
   retry number |
   settings string { active | init-strings string }
   speed number
   }
monitor track-ip
   [
      dynamic | ip | ip ip_addr
   [  interval number | threshold number | weight number ]
   ]
mtrace |
mtu number |
nat |
nsgp [ enforce-ipsec ] |
pbr [ string ] |
phy
   {
   auto |
   full { 10mb | 100mb } |
   half { 10mb | 100mb } |
   holddown number |
   link-down
   } |
port port_name |
pmtu ipv4 |
protocol
   {
   ospf |
   rip [ neighbor ip_addr ] |
   igmp
      [
      config |
      group [ ip_addr [ source ] [ all ] ] |
      source |
      statistic [ all ] |
      ]
   pim [ statistics ]
   } |
proxy dns |
route |
route-deny |
tag id_num zone zone |
vip ip_addr [ + ] port_num [ name_str ip_addr [ manual ] ] |
webauth [ ssl-only ] |
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 (ADSL Interface)

set interface *interface*
backup
{
activation-delay *number* |
auto |
deactivation-delay number |
interface *interface* type { route vroute *string ip_add/mask* | track-ip | tunnel-if }
} |
bandwidth
{ egress mbw *number* | ingress mbw *number* } |
description *string* |
dhcp client
[enable |
settings
{
admin-preference *number* |
autoconfig |
lease *number* |
server *ip_addr* |
update-dhcpserver |
vendor *string*
}
] |
ip { *ip_addr/netmask* | manageable } |
manage
[
ident-reset |
mtrace |
ping |
snmp |
ssh |
ssl |
telnet |
web
]
manage-ip *ip_addr* |
monitor track-ip [dynamic | ip [*ip_addr*] | threshold *number* | weight *number*] |
mtrace |
mtu *number* |

```
pbr [ name_str ] |
phy operating-mode
  {
  adsl2 |
  adsl2plus |
  ansi-dmt |
  auto |
  glite |
  itu-dmt
  } |
pmtu ipv4 |
protocol igmp { host | router } |
proxy dns |
pvc vpi_num vci_num |
    [
    mux { vc | llc }
      [ protocol { routed | bridged } ] |
    qos
      { ubr | cbr pcr_num cdvt_num | vbr-nrt mbs_num scr_num pcr_num
      cdvt_ num } ] |
    zone zone_name
    ]  |
webauth [ ssl-only ] |
webauth-ip ip_addr |
zone zone_name
```

set (basic rate interface, bri0/0, bri1/0, or bri2/0)

```
set interface interface
  {
  alternative-number string |
  backup
    {
    activation-delay number |
    auto |
    deactivation-delay number |
    interface interface type{ route vrouter string ip_add/mask | track-ip | tunnel-if }
    } |
  bri-options
    {
    idle-cycle-flag { flags | ones } |
    loopback { local | remote }
    } |
  dialer-enable |
  disable |
  encap { mlppp | ppp } |
  hold-time { down | up } |
  idle-time number |
  interval number |
  isdn
    {
    calling-number string |
    leased-line 128Kbps |
    send-complete |
    spid1 string |
    spid2 string |
    switch-type { att5e | etsi | ins-net | ni1 | ntdms100 } |
```

t310-value *number* |
tei-negotiation { first-call | power-up }
} |
load-threshold *number* |
monitor track-ip
[
dynamic |
ip [*ip_addr* [interval *number* | threshold *number* | weight *number*]]
threshold *number* |
weight *number* |
]
mtu *number* |
pmtu ipv4 |
primary-number *string* |
proxy dns |
retry *number* |
zone *zone* |
}

set (Cisco HDLC Encapsulation for WAN Interfaces)

set interface *interface*
{
encap cisco-hdlc |
ip unnumbered interface src *interface* |
ip { *ip_add mask* | manageable } |
keepalives
{
interval *seconds* |
down-count *number* |
up-count *number*
}

set (dot1x)

set interface interface dot1x
[
auth-server *string* |
control-mode { virtual | interface } |
max-user *number* |
port-control {force-unauthorized | auto } |
reauth-period *number* |
retry [count | period] |
silent-period *number*
]

set (E1 Interfaces)

set interface *interface*
{
clocking { external | internal } |
e1-options
{
bert-algorithm *name_str* |
bert-error-rate *rate* |
bert-period *seconds* |
fcs { 16 | 32 } |
framing { g704 | g704-no-crc4 | unframed } |
idle-cycle-flag { flags | ones } |
invert-data |
loopback { local | remote } |
start-end-flag { filler | shared } |
timeslots *timeslots* 2-32
} |
hold-time { down *milliseconds* | up *milliseconds* }
}

set (Frame Relay)

set interface *interface*
{
encap frame-relay |
ip unnumbered interface src *interface* |
frame-relay
{
lmi
{
n391-dte *number* |
n392-dte *number* |
n393-dte *number* |
t391-dte *seconds* |
no-keepalive |
type { ansi | itu }
}
} |
}
set interface *subinterface*
{
frame-relay
{
dlci *id_num* |
inverse-arp |
} |
zone *zone* |
ip *ip_add*
}

set (Multilink Frame Relay)

set interface *bundle*
{
bundle-ID *string* |
drop-timeout *milliseconds* |
encap mlfr-uni-nni |
frame-relay lmi
{
n391-dte *number* | n392-dte *number* | n393-dte *number* | t391-dte *seconds* |
no-keepalive |
type { ansi | itu }
} |
minimum-links *number* |
zone *zone* |
ip *ip_add*
}
set interface *bundle_subinterface*
{
frame-relay
{
dlci *id_num* |
inverse-arp |
} |
zone *zone*
}
set interface *interface*
{
bundle *bundle* |
mlfr-uni-nni
{
acknowledge-retries *number* |
acknowledge-timer *milliseconds* |
fragment-threshold *bytes* |
hello-time *milliseconds*
}
}

set (PPP)

set interface *interface*
{
encap ppp |
ip { manageable | *ip_add* | unnumbered interface src *interface* }
keepalives
{
interval *seconds* |
down-count *number* |
}
}

set (Multilink PPP)

set interface *bundle*
{
drop-timeout *milliseconds* |
encap mlppp |
fragment-threshold *bytes* |
minimum-links *number* |
mrru *bytes* |
short-sequence |
zone *zone*
}
set interface *interface*
{
bundle *bundle* |
encap ppp |
ip unnumbered interface *src interface* |
keepalives
{
interval *seconds* |
down-count *number* |
}
}

set (V.92 Modem Interface)

set interface *interface*
{
modem idle-time *number* |
modem interval *number* |
modem retry *number* |
modem settings *name_str* { active | init-strings *name_str* }
modem speed *number* |
modem isp-failover
{ holddown *number* | type { route | track-ip | vpn } vrouter *name_str*} |
modem isp *name_str*
{
account login *name_str* password *pass_str* |
primary-number *string* [alternative-number *string*]
priority *number*
}
}

set (Serial Interfaces)

```
set interface interface
    {
    disable |
    hold-time { down milliseconds | up milliseconds } |
    encapsulation frame-relay |
    serial-options
      {
      clock-rate rate |
      clocking-mode { dce | internal | loop } |
      dce-options
        {
        cts { assert | de-assert | normal } |
        dcd { assert | de-assert | normal } |
        dce-loopback-override |
        dsr { assert | de-assert | normal } |
        dtr { ignore | normal | require } |
        ignore-all |
        rts { ignore | normal | require } |
        tm { ignore | normal | require }
        } |
      dte-options
        {
        cts { ignore | normal | require } |
        dcd { ignore | normal | require } |
        dsr { ignore | normal | require } |
        dtr { ignore | normal | require } |
        ignore-all |
        rts { assert | de-assert | normal } |
        tm { ignore | normal | require }
        } |
      encoding { nrz | nrzi } |
      loopback { dce-local | local | remote } |
      transmit-clock [ invert ]
      }
```

set (T1 Interfaces)

set interface *interface*
{
backup
{
activation-delay *number* |
auto |
deactivation-delay *number* |
interface *interface* type{ route vrouter *string ip_add/mask* | track-ip | tunnel-if }
} |
disable |
clocking { external | internal } |
hold-time { down *milliseconds* | up *milliseconds* } |
t1-options
{
bert-algorithm *name_str* |
bert-error-rate *rate* |
bert-period *seconds* |
buildout { 0-132 | 133-265 | 266-398 | 399-531 | 532-655 } |
byte-encoding { nx56 | nx64 } |
fcs { 16 | 32 } |
framing { esf | sf } |
idle-cycle-flag { flags | ones } |
invert-data |
line-encoding { ami | b8zs } |
loopback { local | payload | remote } |
remote-loopback-respond |
start-end-flag { filler | shared } |
timeslots *timeslots*
}

set (T3 Interfaces)

set interface *interface*
{
disable |
clocking { external | internal } |
hold-time { down *milliseconds* | up *milliseconds* } |
t3-options
{
bert-algorithm *name_str* |
bert-error-rate *rate* |
bert-period *seconds* |
cbit-parity |
compatibility-mode
{
adtran subrate *rate* |
digital-link subrate *rate* |
kentrox subrate *rate* |
larscom subrate *rate* |
verilink subrate *rate*
} |
fcs { 16 | 32 } |
feac-loop-respond
idle-cycle-flag { flags | ones } |
long-buildout
loopback { local | payload | remote } |
payload-scrambler |
start-end-flag { filler | shared }
}

set (Wireless Interfaces)

set interface *interface*
{
shutdown
wlan { 0 | 1 | both }
}

set (Subinterfaces)

set interface *interface*.*id_num*
{
encap pppoe |
tag number zone *zone*
}

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
{
custom *id_num* { integer *number* | ip *ip_addr* | string *string* } |
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
{
enable |
settings
{
autoconfig |
lease *number* |
server *ip_addr* |
update-dhcpserver |
vendor *id_str*
}
}

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 (Loopback Interface)

set interface *interface* loopback-group *interface*

set (Monitoring)

set interface *interface* monitor
{
interface *interface* [weight *number*] |
threshold *number* [action { down | up } { logically | physically }] |
track-ip
[
dynamic |
ip [*ip_addr*] |
threshold *number* |
weight *number*
] |
zone *zone* [weight *number*]
}

set (BGP)

set interface *interface* protocol bgp

set (OSPF)

set interface *interface* protocol ospf
{
area { *ip_addr* | *number* } |
authentication
{
active-md5-key-id *id_num* |
md5 key_str [key-id *id_num*] |
password *pswd_str*
} |
cost *number* |
dead-interval *number* |
enable |
hello-interval *number* |
ignore-mtu |
link-type { p2mp | p2p } |
neighbor-list *number* |
passive |
priority *number* |
reduce-flooding |
retransmit-interval *number* |
transit-delay *number*
}

set (RIP)

set interface *interface* protocol rip
[
authentication
{
active-md5-key-id *id_num* |
md5 key_str [key-id *id_num*] |
password *pswd_str*
} |
enable |
metric *number* |
neighbor { *ip_addr* } |
passive-mode |
receive-version { v1 | v1v2 | v2 } |
route-map *name_str* |
send-version { v1 | v1v2 | v2 } |
split-horizon [poison-reverse]
summary-enable
]

set (IGMP Host)

set interface *interface* protocol igmp host
set interface *interface* protocol igmp
{
enable |
host |
join-group *ip_addr* |
no-check-router-alert |
no-check-subnet |
router |
static-group *ip_addr*
}

set (IGMP Router)

set interface *interface* protocol igmp router
set interface *interface* protocol igmp
{
accept { hosts *id_num* | groups *id_num* | routers *id_num* } |
enable |
join-group *ip_addr* |
last-member-query-interval *number* |
leave-interval *number* |
no-check-router-alert |
no-check-subnet |
proxy [always] |
query-interval *number* |
query-max-response-time *number* |
static-group *ip_addr* |
version { 1 | 2 }
}

set (IRDP)

set interface *interface* protocol irdp
{
ip_addr { advertise | preference *number* }
accept-anonymous-solicitation
broadcast-address
enable
init-adv-interval *seconds*
init-adv-packet *seconds*
lifetime *seconds*
max-adv-interval *upper_limit*
min-adv-interval *lower_limit*
response-delay *seconds*
}

set (PIM)

set interface *interface* protocol pim
[
boot-strap-border |
dr-priority *number* |
enable |
hello-interval *number* |
join-prune-interval *number* |
neighbor-policy *number*
]

set (Policy-Based Routing)

set interface interface pbr *pbr_policy_name*

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* | unnumbered interface *interface* } |
loopback-group |
manage-ip *ip_addr* |
mip *ip_addr* host *ip_addr*
[netmask *mask* [vrouter *name_str*]] |
mtrace |
mtu *number* |
nhtb *ip_addr* vpn *tunn_str* |
protocol
{
bgp |
ospf [demand-circuit] |
rip [demand-circuit] |
igmp |
pim
} |
proxy dns |
route-deny |
tunnel
{
encap gre [key] |
keep-alive [interval *number* | threshold *number*] |
local-if interface dst-ip *ip_addr*
} |
zone *name_str*
}

NOTE: Use the IP option only after adding the tunnel to a specific zone.

Keywords and Variables

Variable Parameter

get interface *interface* | *subinterface* | *bundle* | *bundle_subinterface* ...
set interface *interface* | *subinterface* | *bundle* | *bundle_subinterface* ...

interface	The name of the interface. All WAN interfaces on SSG devices, including serial, T1/E1, and T3, are named serial*n1*/*n2*, where *n1* is the slot number in the SSG chassis that is occupied by the Physical Interface Module (PIM), and *n2* is the physical port on the PIM. For MLFR and MLPPP, you configure and add physical interfaces to the bundle interface.
src interface	The name of the source interface to which an unnumbered interface is assigned an IP address. You can configure an unnumbered interface to use a source interface when the unnumbered interface does not work.
subinterface	(Frame Relay only) The name of a virtual interface that is associated with a physical interface. You can create multiple subinterfaces on a physical interface. Subinterface names consist of the physical interface name, followed by a subinterface identification number, for example, serial1/1.1 or serial1/1.2.
bundle	(MLFR and MLPPP only) The name of the bundle interface. Bundle interface names consists of ml, followed by an identification number. For example, bundle interface names can be ml1, ml2, and so on.
bundle_subinterface	(MLFR only) The name of a virtual interface that is associated with a bundle interface. You can create multiple subinterfaces on a bundle interface. Subinterface names consist of the bundle interface name, followed by a subinterface identification number, for example, ml1.1 or ml1.2.

Example: The following command specifies the IP address of a remote gateway peer (1.1.1.25) for the serial interface in port 0 of the PIM in slot 1:

set interface serial1/0 gateway 1.1.1.25

account login

set interface *interface* modem isp *name_str* account login *string* password *pswd_str*

account login	Specifies the login name (*string*)and account password (*pswd_str)* for the ISP account.

Example: The following command configures the login juniper and the password bodie45 for the ISP account *isp1*:

set interface serial1/0 modem isp isp1 account login juniper password bodie45

alternative-number

set interface *interface* alternative-number *string*
unset interface *interface* alternative-number *string*

alternative-number *string*	Specifies the remote destination to call. If the primary number is not connected, **alternative-number** is used. The **alternative-number** is a string from 1 to 15 characters.

association

set interface *interface* association [*mac_addr*]

association	Displays wireless clients associated to the wireless interface. To see more information about a particular client, specify its MAC address with the optional *mac_addr*.

backup

set interface *interface* backup { ... }
unset interface *interface* backup { ... }
exec backup interface *interface* { failover \| revert }

backup — Specifies the settings for the backup interface.

- **activation-delay** *number* Specifies the number of seconds to wait after the primary interface goes down and the backup interface is activated. The range is 1-60 and the default is 30.
- **auto** Configures the backup interface to fail over or revert to the primary interface automatically.
- **deactivation-delay** *number* Specifies the number of seconds to wait to bring down the backup interface after the primary interface is up. The range is 1-60 and the default is 30.
- **interface** *interface* Specifies the interface that acts as backup interface. Select the method to determine if the primary interface is unavailable.
  - **type** Specifies the type of event to trigger failover or recover.
    - **route vrouter** *string ip_add/mask* Enables the backup interface if the preconfigured route becomes unreacheable through the interface.
    - **track-ip** Enables the backup interface when certain IP addresses become unreachable through the interface.
    - **tunnel-if** Enables the backup interface when certain VPN tunnels on the interface become unreachable through VPN tunnel monitoring.
- failover Forces the interface to failover to the backup interface.
- revert Forces the interface to revert to the primary interface.

Example: The following command specifies the serial2/0 as backup interface for bri1/0. Once the route 10.10.10.10/24 in vrouter trust-vr is deactivated, the failover takes place.

set interface bri1/0 backup interface serial2/0 type route vrouter trust-vr 10.10.10.10/24

bandwidth

set interface *interface* bandwidth { egress mbw *number* | ingress mbw *number }*
unset interface *interface* bandwidth

bandwidth

- **egress** The maximum bandwidth in kilobits per second for all traffic traversing the egress interface.
- **ingress** The maximum bandwidth in kilobits per second for all traffic traversing the ingress interface.

Example: The following command specifies bandwidth of 10,000 kilobits per second for interface *ethernet4*:

set interface ethernet4 bandwidth egress mbw 10000
set interface ethernet4 bandwidth ingress mbw 10000

bert-test

exec interface *interface* bert-test [start | stop]

bert-test — Starts or stops bit error rate testing on the specified interface.

bri-options

get interface *interface* bri-options
set interface *interface* bri-options { ... }
unset interface *interface* bri-options { ... }

bri-options

- **idle-cycle-flag** Specifies the value the BRI interface transmits during its idle cycles. Select **ones** (0xFF) or **flags** (0x7E) to configure the value the BRI interface transmits during idle-cycles in order to keep the line up. The default is **ones** (0xFF).
- **loopback** Specifies the maximum bandwidth in kilobits per second for all traffic traversing the ingress interface. Loopback mode is disabled by default.
  - **remote** Received data is looped back to the S interface. The D-channel information received from the line card is output to the S interface transparently.
  - **local** Performs complete system diagnostics. The transmitted data is looped back to the receiver through the S interface. (The pin-out of the external loop cable is pin 3 <-> pin 4 and pin 5 <-> pin 6).

Example: The following command specifies remote loopback mode:

set interface bri1/0 lri-options loopback remote

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

bundle

set interface *interface* bundle *bundle*

bundle	(For multilink interfaces only) Adds the physical link *interface* to the multilink interface *bundle*.

bundle-ID

set interface *bundle* bundle-ID *string*

bundle-ID	Specifies an identifier for the bundle interface. If you do not specify a bundle ID, the bundle interface name is used.

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) with a unicast MAC destination address 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 this feature is disabled.) Executing the **unset interface** *interface* **bypass-non-ip** command drops all the non-IP packet with unicast MAC destination addresses, but non-IP packets with multicast MAC addresses are still passed through.

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 this feature is disabled.) Executing the **unset interface** *interface* **bypass-non-ip-all** drops all non-IP packets, regardless of the MAC destination address.

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.

cisco-hdlc

get interface *interface* cisco-hdlc

cisco-hdlc	Shows the statistics and configuration information for an interface configured for Cisco High-Level Data Link Control protocol.

clocking

set interface *interface* clocking external | internal

clocking	Specifies the clocking source for T1/E1 or T3 lines. You can specify one of the following options: ■ **external** Specifies that clocking is provided by the DCE (loop timing). ■ **internal** Specifies that clocking is provided by the SSG device’s own system clock. This is the default.

description

set interface *interface* description *string*
unset interface *interface* description

description	Adds a description (*string*) to an interface.

dhcp client

set interface *interface* dhcp client
{
enable |
settings
{
admin-preference *number* |
autoconfig |
lease *number* |
server *ip_addr* |
update-dhcpserver | vendor *id_str* } }

dhcp client — Configures an interface for DHCP client services.

- **enable** Enables DHCP client services for the interface.
- **settings** Configures DHCP parameters for the interface.
  - **admin-preference** *number*
  - **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.

Example 1: The following command configures interface *ethernet3* to perform automatic DHCP configuration after device power-up:

set interface ethernet3 dhcp client settings autoconfig

Example 2: 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 security 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 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 security 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.

- **custom** *id_num* Creates a user-defined value for configurations where the predefined server options (listed below) do not suffice, and you need to define custom DHCP server options. For example, certain VoIP (Voice-over IP) configurations require transmission of extra configuration information, which is not currently supported by predefined server options. In such cases, you must define suitable custom options.
  - **string** *string* Specifies a character string.
  - **ip** *ip_addr* Specifies an IP address.
  - **integer** *number* Specifies an integer value.
- **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

dialer-enable

set interface *interface* dialer-enable
unset interface *interface* dialer-enable

dialer-enable	Sets the ISDN BRI interface to enable dialing. The BRI interface acts as a dialer interface. It has two dialer-pool members by default, the two B-channels. The BRI interface does not enable dialing by default.

dip

set interface *interface* [ext ip *ip_addr/mask*] dip *id_num* *ip_addr1* [*ip_addr2*] [fix-port | incoming]
set interface *interface* [ext ip *ip_addr/mask*] 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. The security device can use the pool to dynamically or deterministically allocate source addresses when the device applies source address translation (NAT-src) to packets traversing the specified interface. This is useful when you need to translate nonroutable local IP source addresses into routable addresses for outgoing packet. The keywords and variables for the **dip** option are as follows:

- *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 a range of addresses.) The second IP address *ip_addr2* represents the end of the IP address range.
- [ext ip *ip_addr/mask*] is the extended interface IP address.
- **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.
- **incoming** Creates a DIP address pool for dynamically allocating destination addresses. The name of the DIP pool can be DIP(*id_num*) for a user-defined DIP, or DIP(*interface*) for an interface DIP. The DIP address pool resides in the Global security zone. You can use such address entries as destination addresses in policies, together with the services H.323, SIP, or VoIP (Voice-over IP), to support incoming calls.

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

interface-ip incoming Designates addresses derived from the interface IP address range for dynamically allocating destination addresses to incoming packets.

Example 1: The following commands allow local hosts in a nonroutable subnet to communicate over a public WAN infrastructure. The security device uses a DIP pool to dynamically allocate routable source addresses to packets sent from the local hosts to remote hosts.

- Local unroutable subnet 10.1.23.1/24
- Remote unroutable subnet 10.100.2.75/24
- DIP ID number 10, with address range from 2.1.10.2 through 2.1.10.36

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 10.1.23.1/24
set interface ethernet3 zone untrust
set interface ethernet3 ip 2.1.10.1/24
set interface ethernet3 dip 10 2.1.10.2 2.1.10.36
set address trust Local_Hosts 10.1.23.1/24
set address untrust Remote_Hosts 10.100.2.75/24
set policy from trust to untrust Local_Hosts Remote_Hosts http nat dip 10 permit

Example 2: The following commands use DIP in an H.323 VoIP configuration.

- Creates a pool of DIP addresses (identified by ID 5) containing addresses 1.1.1.12 through 1.1.1.150 inclusive. The device can use addresses in this DIP pool as incoming destination addresses (or as outgoing source addresses).
- Creates a policy that allows outgoing H.323 requests, using DIP addresses for source addresses.
- Creates a policy that allows incoming H.323 requests, using DIP addresses for destination addresses.

set interface ethernet7 ip 1.1.1.1/24
set interface ethernet7 dip 5 1.1.1.12 1.1.1.150 incoming
set policy from trust to untrust any any h.323 nat src dip 5 permit
set policy from untrust to trust any dip(5) h.323 permit

disable

set interface *interface* disable

disable	Disables the interface. WAN interfaces are enabled by default.

dot1x

clear interface *interface* dot1x statistics
get interface *interface* dot1x [...]
set interface *interface* dot1x [...]
unset interface *interface* dot1x [...]

auth-server *string*	Specifies a predefined server as the authentication server for the interface.
control-mode	Specifies whether MAC address-based authentication is performed on devices connected to the interface. ■ interface: MAC addresses of devices connected to the interface are not authenticated. Use this option if only one trusted device is connected to the interface. ■ virtual: MAC addresses of devices connected to the interface are authenticated. Packets from devices with unauthorized MAC addresses are dropped. This mode is the default for an interface. Wireless interfaces use only virtual mode.
max-user *number*	Maximum number of users that require 802.1X authentication on an interface. This option is available only if virtual mode (using the **set interface interface dot1x control-mode** command) is configured. The maximum number of users is 1 through 256. By default, the maximum number of users is 16 for wired interfaces and 256 for wireless interfaces.
port-control	Specifies the 802.1X authentication state of the interface: ■ **auto**: Allows authentication to proceed normally, as defined by 802.1X. This option is the default for an interface. ■ **force-unauthorized**: Forces the interface to block all traffic and ignore all attempts by clients to authenticate.
reauth-period *number*	Amount of time the security device waits before attempting reauthentication of clients. By default, the security device waits 3600 seconds (1 hour) before attempting client reauthentication. The value range is 0 through 86400 seconds (24 hours). Setting the value to 0 disables reauthentication. Use the unset interface interface_name dot1x reauth-period to revert to the default value.
retry	Enables retransmission of EAP requests to a client if it does not respond. By default, retransmission is enabled. If the maximum number of retransmissions is reached, the client's authenticated session is terminated, and authentication fails. Optionally, set the maximum number of EAP requests that are retransmitted and the time that elapses between retransmissions to the client if it does not respond. ■ **count** *number*: Maximum number of EAP requests from 1 through 16. The default value is 3. ■ **period** *number*: Period between retransmissions in the value range of 1 through 120 seconds. The default value is 3 seconds. Use the **unset interface** *interface_name* **dot1x retry** [**count** | **period**] to revert to the default value.

silent-period *number*	Amount of time the security device remains silent after authentication has failed. During the silent period, the security device does not initiate or respond to any client authentication requests. By default, when authentication fails, the security device is silent for 5 seconds. The authentication retry count resets to zero (0). The silent period is a value from 0 through 3600 seconds (1 hour). If you specify a zero value, the 802.1X authentication state remains unauthorized after the retry fails. Use the **unset interface** *interface_name* **dot1x silent-period** to revert to the default value.
statistics	Displays or clears statistics for an interface on which 802.1X is enabled.

Example: The following configuration scenario illustrates a network setup for a hub with attached clients connected to the security device with the following parameters:

- Hub connected to Ethernet2 interface
- Ethernet3/1 interface bound to Trust zone with an IP address of 10.1.40.3/24
- RADIUS server named radius1 (10.1.1.200) connected to Ethernet3 interface to authenticate users with 802.1X, using port 1812 as the authentication port and secret of mysecret

```
set interface ethernet2 dot1x
set interface ethernet2 dot1x control-mode virtual

set interface ethernet3 zone trust
set interface ethernet3 ip 10.1.1.10/24

set auth-server radius1 account-type 802.1x
set auth-server radius1 type radius
set auth-server radius1 radius port 1812
set auth-server radius1 radius secret mysecret
set auth-server radius1 server-name 10.1.1.200

set interface ethernet2 dot1x auth-server radius1
```

drop-timeout

set interface *bundle* drop-timeout *milliseconds*

drop-timeout	(For multilink bundle interfaces only) Specifies the drop timeout in milliseconds. The drop timeout provides a recovery mechanism if individual links in the multilink bundle drop one or more packets. The default is 0, which means that drop timeout is disabled. Specify a value between 0-127 milliseconds.

e1-options

set interface *interface* e1-options ...

e1-options

Specifies options for an E1 interface. You can specify the following:

- **bert-algorithm** Sets the bit error rate testing (BERT) algorithm for the interface. The algorithm is the pattern to send in the bitstream. You can specify one of the following options:
  - **all-ones-repeating** Repeating one bits.
  - **all-zeros-repeating** Repeating zero bits.
  - **alternating-double-ones-zeros** Alternating pairs of ones and zeroes.
  - **alternating-ones-zeros** Alternating ones and zeroes.
  - **pseudo-2e10** Pattern is 2^10-1.
  - **pseudo-2e11-o152** Pattern is 2^11-1 (per O.152 standard).
  - **pseudo-2e15-o151** Pattern is 2^15-1 (per O.152 standard). This is the default.
  - **pseudo-2e17** Pattern is 2^17-1.
  - **pseudo-2e18** Pattern is 2^18-1.
  - **pseudo-2e20-o151** Pattern is 2^20-1 (per O.151 standard).
  - **pseudo-2e20-o153** Pattern is 2^20-1 (per O.153 standard).
  - **pseudo-2e21** Pattern is 2^21-1.
  - **pseudo-2e22** Pattern is 2^22-1.
  - **pseudo-2e23-o151** Pattern is 2^23 (per O.151 standard).
  - **pseudo-2e25** Pattern is 2^25-1.
  - **pseudo-2e28** Pattern is 2^28-1.
  - **pseudo-2e29** Pattern is 2^29-1.
  - **pseudo-2e3** Pattern is 2^3-1.
  - **pseudo-2e31** Pattern is 2^31-1.
  - **pseudo-2e32** Pattern is 2^32-1.
  - **pseudo-2e4** Pattern is 2^4-1.
  - **pseudo-2e5** Pattern is 2^5-1.
  - **pseudo-2e6** Pattern is 2^6-1.
  - **pseudo-2e7** Pattern is 2^7-1.
  - **pseudo-2e9-o153** Pattern is 2^9-1 (per O.153 standard).
  - **repeating-1-in-4** One bit in 4 is set.
  - **repeating-1-in-8** One bit in 8 is set.
  - **repeating-3-in-24** Three bits in 24 are set.
- **bert-error-rate** Sets the bit error rate (BER) to use in BERT. This can be an integer from 0 to 7, which corresponds to a BER from 10^{-0} (1 error per bit) to 10^{-7}. The default is 0.
- **bert-period** Sets the length of the BERT, in seconds. The default is 10. Specify a value between 1 and 240 seconds.
- **fcs** Specifies the number of bits in the frame checksum. You can specify one of the following:
  - **16** 16 bits. This is the default.
  - **32** 32 bits.

- **framing** Sets the framing mode for the E1 line. You can specify the following:
  - **g704** G704 mode with cyclic redundancy check 4 (CRC 4). This is the default.
  - **g704-no-crc4** G704 mode without CRC4.
- **idle-cycle-flag** Sets the value to transmit in idle cycles. You can specify one of the following:
  - **flags** Transmit 0x7E in idle cycles. This is the default.
  - **ones** Transmit 0xFF (all ones) in idle cycles.
- **invert-data** Specifies data inversion. Data inversion is normally used only in alternate mark inversion (AMI) mode. By default, this is not set.
- **loopback** Specifies loopback mode. By default, no loopback mode is set. You can specify one of the following:
  - **local** Local loopback.
  - **remote** Remote loopback.
- **start-end-flag** Sets the start and end flags on transmission. You can specify one of the following:
  - **filler** Send two idle cycles between start/end flags. This is the default.
  - **shared** Share start/end flags on transmit.
- **timeslots** *timeslots* Specifies the number of time slots allocated to a fractional E1 interface. By default, all time slots are active. Specify values from 2 to 32. Use hyphens to specify a range. Use commas (with no spaces before or after) to separate individual time slots or ranges. For example, you can specify the following: 3-5,9,22-24,28.

encap

set interface *interface* encap { cisco-hdlc | frame-relay | mlfr-uni-nni | mlppp | ppp }
set interface *bundle* encap { mlfr-uni-nni | mlppp }

encap	Specifies the type of encapsulation to perform, when the subinterface is untagged. An untagged interface does not use a VLAN tag to identify a VLAN for an subinterface. Instead, it binds the subinterface to a particular defined PPPoE instance. Thus, by hosting multiple subinterfaces, a single physical interface can host multiple PPPoE instances. You can configure each instance to go to a specified AC (Access Concentrator), thus allowing separate entities such as ISPs to manage the PPPoE sessions.
cisco-hdlc	Sets Cisco High-Level Data Link Control (Cisco HDLC) encapsulation on the specified interface.
frame-relay	Sets Frame Relay encapsulation on the specified interface.
mlfr-uni-nni	(For Frame Relay multilink bundle interfaces only) Sets Multilink Frame Relay User-to-Network Interface (UNI) encapsulation, based on Frame Relay Forum Multilink Implementation Agreement FRF.16, on the specified interface.
mlppp	(For MLPPP bundle interfaces only) Sets Multilink Point-to-Point Protocol on the specified interface.
ppp	Sets Point-to-Point Protocol (PPP) encapsulation on the specified interface.

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 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 dialer-enable on page 275. ■ **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 10.1.10.110

fragment-threshold

set interface *bundle* fragment-threshold *bytes*

fragment-threshold	(For MLPPP bundle interfaces only) Specifies the maximum size, in bytes, for packet payloads transmitted across the individual links within the multilink circuit. The threshold value affects the payload only; it does not affect the MLPPP header. The default value is 0 bytes (disabled). Specify a value between 128-16320 bytes.

frame-relay

get interface *interface* frame-relay
set interface *interface* frame-relay ...

frame-relay	For the **get** command, shows the statistics and configuration information for an interface configured for Frame Relay or Multilink Frame Relay. The interface can be a WAN interface, a WAN subinterface, a bundle interface, or a bundle subinterface.
dlci *id_num*	(For Frame Relay subinterfaces only) Configures the data link connection identifier (DLCI) for a permanent virtual circuit (PVC) for Frame Relay and Multilink Frame Relay user-to-network interface (UNI) encapsulations. Specify a value between 16 and 1022.
inverse-arp	(For Frame Relay subinterfaces only) Configures the router to respond to inverse Frame Relay Address Resolution Protocol (ARP) requests by providing IP address information to the requesting router at the other end of the Frame Relay PVC.
lmi	Sets the type of Local Management Interface (LMI) packets used for keepalives and keepalive settings. You can specify the following:

- **n391-dte** *number* Specifies the data terminal equipment (DTE) full status polling interval. The DTE sends a status inquiry to the data circuit-terminating equipment (DCE) at the interval specified by **t391-dte**. **n391-dte** specifies the frequency at which these inquiries expect a full status report; for example, a **n391-dte** value of 10 would specify a full status report in response to every tenth inquiry. The intermediate inquiries ask for a keepalive exchange only. The range is from 1 through 255, with a default value of 6.
- **t391-dte** *seconds* Specifies the DTE keepalive timer, which is the period at which the DTE sends out a keepalive response request to the DCE and updates status depending on the DTE error-threshold value. The range is from 5 through 30 seconds, with a default value of 10 seconds.
- **n392-dte** *number* Specifies the DTE error threshold, which is the number of errors required to bring down the link, within the event-count specified by **n393-dte**. The range is from 1 through 10, with a default value of 3.
- **n393-dte** *number* Specifies the DTE monitored event-count. The range is from 1 through 10, with a default value of 4.
- **no-keepalive** Disables the sending of keepalives on the interface.
- **type** Specifies the type of LMI packets for keepalives. You can specify one of the following:
  - **ansi** Specifies ANSI T1.617 Annex D LMIs.
  - **itu** Specifies ITU Q933 Annex A LMIs.

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

hold-time

get interface *interface* hold-time
set interface *interface* hold-time { down | up }
uset interface *interface* hold-time { down | up }

hold-time	Specifies the link state hold time or how much time can pass before the device considers the interface connection to be up or down. The range is 0 - 65534 (milliseconds). The default value for **up/down** time is 0 (no damp).

- **up** Configures the hold-time period when an interface goes from up to down, it is not advertised as being down until it has remained down for the specified **up** period.
- **down** Configure the hold-time period when an interface goes from down to up, it is not advertised as being up until it has remained up for the specified **down** period.

idle-time

set interface *interface* modem idle-time *number*
set interface *interface* idle-time *number*
unset interface *interface* modem idle-time *number*
uset interface *interface* idle-time *number*

idle-time	Specifies the number of seconds that elapse with no traffic on the dial-up connection before the security device disconnects the modem. The default is 1 minute. A value of 0 means the modem never disconnects, even if there is no traffic on the dial-up connection.

Example: The following command sets an idle time of 12 seconds:

set interface serial1/0 modem idle-time 12

Example: The following command sets an idle time of 12 seconds for the basic rate interface (ISDN):

set interface bri1/0 idle-time 12

interval

set interface *interface* modem interval *number*
set interface *interface* interval *number*
unset interface *interface* modem interval *number*
unset interface *interface* interval *number*

interval	Specifies the seconds (*number*) between dial-up retries. Valid interval range is 1-60 seconds and the default is 60 seconds.

Example: The following command sets a dial-up interval of 45 seconds:

set interface serial1/0 modem interval 45

Example: The following command sets a dial-up interval of 45 seconds for the basic rate interface (ISDN):

set interface bri1/0 interval 45

ip

set interface *interface* ip *ip_addr/mask* [secondary]
set interface *interface* ip unnumbered interface *interface2*
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. Use the unnumbered option if the tunnel interface does not need to support policy-based NAT, and if your configuration does not require the tunnel interface to be bound to a tunnel zone. The **unnumbered** option specifies that the tunnel interface is unnumbered. It does not have an IP address, but instead borrows the IP address from another interface (interface2). The other interface is bound to the same security zone. **Warning:** RIP is *not* supported over unnumbered tunnel interfaces. All interfaces that use RIP protocol must be numbered. Any attempt to configure and run an unnumbered interface using RIP may lead to unpredictable routing failure.

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 unnumbered interface

set interface *interface* ip unnumbered interface *src interface*

ip unnumbered interface	Enables the local address to be derived from a source interface (*src interface*) which has been configured with an IP address.

isdn

get interface *interface* isdn { ... }
set interface *interface* isdn { ... }
unset interface *interface* isdn { ... }

q921	Displays information on the Q921 protocol or responses exchanged during peer-to-peer communication carried over the D channel. ■ statistics Shows the number of transmitted and received frame types. ■ status Displays the Layer 2 status, TEI state and the TEI assigned value.
q931	Displays information on the Q931 protocol. ■ statistics Shows the number of transmitted and received message types. ■ status Displays the number of Active calls.
calling-number *string*	Supplies the ISDN network with a billing number for outgoing calls. The device dials the number, and the switch selects the route. Some networks offer better pricing on calls where the number is presented. When configured, this information is included in the outgoing call setup message.

leased-line 128Kbps	Specifies a layer 3 interface and is predefined for a data rate of 128Kbps. There is is no signalling on the D channel and the leased line is used to deliver data only.
send-complete	Includes send-complete information in the outgoing setup message to indicate that the entire number is included. ISDN switches require this information in certain geographic locations, such as Hong Kong and Taiwan requi. This information element is generally not required in other locations. The default is not set.
spid1*string* spid2 *string*	Specifies the service available to you on the ISDN switch that defines the feature set ordered when you provisioned for the ISDN service. A Service Profile Identifier (SPID) number is usually a seven-digit telephone number with some optional numbers. However, service providers may use different numbering schemes. If you are using a service provider that requires a SPID, your device cannot place or receive calls until it sends a valid, assigned SPID to the service provider when it accesses the ISDN switch to initialize the connection. **Note:** Currently, only the DMS-100 and ni1 ISDN switch types require SPIDs. For the DMS-100 switch type, two SPIDs are assigned, one for each B-channel. Do not specify SPID numbers, if you selected the AT&T 5ESS ISDN switch type. In addition, SPIDs are important at the local access ISDN interface only. Remote routers never receive the SPID number.
switch-type	Specifies the ISDN switch type: ■ **att5e** (AT&T 5ESS) ■ **etsi** (European variants) ■ **ins-net** (NTT INS-NET) ■ **ni1** (National ISDN-1) ■ **ntdms100** (Nortel DMS100) Choose the switch with the help of your Internet Service Provider. Do not change the switch type during operation. The updated switch type will take into effect after the device reboots.
t310-value *number*	Sets the timeout value in seconds if ALERT, CONNECT, DISC, or PROGRESS is not received after a CALL PROC. Then, a DISC is sent to the network side for the duration of the T310 timeout value. The range is 5-100 and the default is 10.
tei-negotiation	Identifies the Terminal Endpoint Identifier (TEI) that connects to the ISDN switch. It's always dynamically assigned by the ISDN switch. Both settings conform to different standards, ANSI & ETSI. The default is **first-call**. ■ **first-call** Enables the device to activate the TEI negotiation when the first call is made. ■ **power-up** Allows the switch to assign TEI once the device boots up. TEI negotiation is useful for switches that may deactivate Layers 1 or 2 when there are no active calls. Typically, this setting is used for ISDN service offerings in Europe and connections to DMS-100 ISDN switches that are designed to initiate TEI negotiation.

isp

set interface *interface* modem isp *name_str* { ... }
unset interface *interface* modem isp *name_str*

isp	Specifies the ISP.

Example: The following command configures the login *juniper* and the password *bodie45* for the ISP *isp1*:

set interface serial1/0 modem isp isp1 account login juniper password bodie45

isp-failover

set interface *interface* modem isp-failover holddown *number*
set interface *interface* modem isp-failover type { route | track-ip | vpn } vrouter *vr_name ipaddr/mask*
unset interface *interface* modem isp-failover holddown
unset interface *interface* modem isp-failover type

isp-failover

Allows you to configure up to four ISPs for failover and dial-up connections. The holddown timer and type arguments can be configured as follows:

- **holddown** *number* specifies the number of seconds to wait before initiating failover. The default value is 30 seconds; however, the valid range is between 1 and 300 seconds. The **unset** command returns the holddown value to the default. Using the **set** command twice overwrites the previous value.
- **type** { ... } **vrouter** *vr_name ip_addr/mask* specifies a route generated by a dynamic routing protocol, such as OSPF or BGP. The security device monitors the status of the interface in the virtual router. this feature is disabled by default.

keepalives

set interface *interface* keepalives { interval *seconds* | down-count *number* | up-count *number* }

keepalives

By default, physical interfaces configured with Cisco-HDLC or PPP encapsulation send keepalive packets at 10-second intervals. You can configure the following keepalive parameters:

- **interval** Specifies the interval at which the interface sends keepalive packets on a link. The default is 10 seconds. Specify a value between 1-32767 seconds.
- **down-count** Specifies the number of successive times that a destination fails to receive keepalive packets before it considers the link to be down. The default is 3 times. Specify a value between 1-255.
- **up-count** (Cisco HDLC encapsulation only) Specifies the number of times that a destination must receive a keepalive packet before it considers the link to be up. The default is 0 (disabled). Specify a value between 1-255.

load-threshold

set interface *interface* load-threshold *number*
unset interface *interface* load-threshold *number*

load-threshold	Sets up the second B channel for bandwidth on demand and if the traffic is greater than the load-threshold (in percentage). The range is from 1-100 and the default is 80.

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 | mtrace | nsmgmt | ping | snmp | ssh | ssl | telnet | web }
unset interface *interface* manage
{ ident-reset | mtrace | 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.
- **mtrace**
- **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 377.
- **mtrace** Enables (or disables) mtrace manageability on the interface. (Mtrace traces a route to the source device using a multicast address.)
- **ping** Enables (or disables) pinging 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

minimum-links

set interface *bundle* minimum-links *number*

minimum-links	(For multilink bundle interfaces only) Sets the minimum number of bundle links that must be up for the bundle to be considered up. The default is 1. You can specify a value between 1-8.

mip

set interface *interface* mip *ip_addr1* host *ip_addr2* [vrouter *vrouter*] [netmask *mask*]
unset interface *interface* mip *ip_addr1* [netmask *mask*]

mip — Defines a Mapped IP (MIP) address for the security interface. The device directs traffic sent to the MIP (*ip_addr1*) to the host with the IP address *ip_addr2*. 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: Be careful to exclude the interface and gateway IP addresses, and any Virtual IP addresses in the subnet from the MIP address range.)**vrouter** *vrouter* Identifies the virtual router containing a route to the host device.
- **netmask** Specifies the range of host IP addresses.

Example: 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 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

mlfr-uni-nni

set interface *interface* mlfr-uni-nni ...

mlfr-uni-nni — (For multilink bundle links only) Configures options for Multilink Frame Relay FRF.16 operations. You can configure the following:

- **acknowledge-retries** *number* Specifies the number of retransmission attempts to be made for consecutive hello or remove-link messages after the expiration of the acknowledgement timer. The default is 2. Specify a value between 1-5.
- **acknowledge-timer** *milliseconds* Specifies the maximum period, in milliseconds, to wait for an add-link, hello, or remove-link acknowledgement.The default is 4 milliseconds. Specify a value between 1-10.
- **fragment-threshold** *bytes* Specifies the maximum size for packet payloads transmitted across bundle links within a multilink circuit. The default is the maximum transmission unit (MTU) of the physical link. Specify a multiple of 64 bytes.
- **hello-timer** *milliseconds* Specifies the rate, in milliseconds, at which hello messages are sent. The default is 10 milliseconds. Specify a value between 1-180.

modem

set interface *interface* modem { ... }
unset interface *interface* modem { ... }

modem	Configures modem settings for the specified interface.

The modem keyword options are as follows.

- **idle-time** *number* Specifies the number of minutes that elapse with no traffic on the dial-up connection before the security device disconnects the modem. The default is 1 minute. A value of 0 means the modem never disconnects, even if there is no traffic on the dial-up connection.
- **interval** *number* Specifies the seconds (*number*) between dial-up retries. The default is 60 seconds. Range is 3-60 seconds.
- **retry** *number* Specifies the number of times ScreenOS dials the primary number, and then the alternative-number, if the line is busy or there is no answer from the ISP. The default is 3 times. The range is 0-10 times.
- **settings** *name_str* { active | init-strings *name_str* } Configures settings for the specified modem or ISP.
- **speed** *number* Specifies the maximum baud rate for the serial link between the device and the modem. The baud rate can be 9600, 19200, 38400, 57600, or 115200 bps. The default is 115200 bps.
- **isp** *name_str* { account login *name_str* password *pass_str* | priority *number* } Configures ISP information.
- **isp-failover** { holddown *number* | type { route | track-ip |vpn } vroute *name_str* }Allows you to configure up to four ISPs for failover and dial-up connections. The holddown timer and type arguments can be configured as follows:
  - **holddown** *number* specifies the number of seconds to wait before initiating failover. The default value is 30 seconds; however, the valid range is between 1 and 300 seconds. The **unset** command returns the holddown value to the default. Using the **set** command twice overwrites the previous value.
  - **type** { ... } **vrouter** *vr_name ip_addr/mask* specifies a route generated by a dynamic routing protocol, such as OSPF or BGP. The security device monitors the status of the interface in the virtual router. this feature is disabled by default.

monitor

set interface *interface* monitor { ... }
unset interface *interface* monitor { ... }

monitor	Configures monitoring for the specified interface.

An interface can monitor objects for any of the following events. Each of these events by itself or in combination can cause the state of the monitoring interface to change from up to down or from down to up.

- Physical disconnection or reconnection
- IP tracking failure or success

  When the tracking of an IP address fails, the device compares the weight assigned to the tracked IP address with the failure threshold for tracked objects. If the number of failures exceed the threshold, the device compares the weight for tracked objects with the failure threshold. If the number of failures exceeds the threshold, the interface changes its state (from up to down, or down to up).

- Failure or success of a monitored interface

  When a monitored interface changes state, the device compares the weight assigned to the monitored interface with the failure threshold for interface monitoring. If the number of failures exceeds the threshold, the interface changes its state (from up to down, or down to up).

- Failure or success of a monitored security zone

  An interface can monitor all the interfaces in any security zone other than its own. For an entire security zone to fail, every interface bound to that zone must fail. As long as one interface bound to a monitored zone is up, the device considers the entire zone to be up. When a monitored zone changes state, the security device compares the weight assigned to the monitored zone with the failure threshold.

The security device uses ping requests to poll the remote device.

The monitoring keyword options are as follows.

- **interface** *interface* [**weight** *number*] Identifies the interface from which the device sends the ping requests, and the relative weight assigned to the interface.
- **threshold** *number* [action { down | up } { logically | physically }] The failure rate at which the interface goes from up to down or down to up.
- **track-ip** Configures the Track IP feature.
  - **dynamic** Enables the Track IP feature.
  - **ip** [*ip_addr*] Identifies the tracked IP address.
  - **threshold** *number* Indicates the number of consecutive failures required to elicit a ping response from a specific IP address required to be considered a failed attempt.
  - **weight** *number* Indicates the weight of the IP address. The weight is the amount that the tracked object failure contributes toward the monitored object failure threshold.
- **zone** *zone* [**weight** *number*] Indicates the weight of the zone.

mrru

set interface *bundle* mrru *bytes*

mrru	(For MLPPP bundle interfaces only) Specifies the maximum packet size, in bytes, that the multilink interface can process. The default is 1500 bytes. Specify a value between 1500-4500.

mtrace

set interface *interface* mtrace
unset interface *interface* mtrace

mtrace	Allows you to do packet tracing from a multicast receiver to a source.

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 800 and 1500.

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 only available when the device is in Route Mode, in which the interfaces have assigned IP addresses.

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 IP address of a remote peer's tunnel interface (*ip_addr*) in the NHTB (next-hop tunnel binding) 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

nsgp

set interface *interface* nsgp [enforce-ipsec]

nsgp	For GPRS systems, enables or disables the exchange of Overbilling Attack information through the specified interface on the security device. You must set an interface on both security devices: the GTP firewall (client) and the Gi firewall (server). The interface for the client and server must have different IP addresses. Also, you can enable NSGP on a physical Ethernet interface only. The **enforce-ipsec** switch sets the interface to only accept incoming connections from an IPSec tunnel.

pbr

set interface *interface* pbr *pbr_policy_name*
unset interface *interface* pbr *pbr_policy_name*

pbr	Enables a Policy-Based Routing (PBR) policy to be bound to the specified interface.If a PBR policy name is not specified, then any declared policy will be used. If no PBR policies exist at the zone or virtual router level, then normal route lookup this performed even though PBR is enabled on the interface. A lack of a PBR policy does not prevent the device from performing packet forwarding.

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 or half duplex (as required by the network device connected 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 hold-down time for the link, in increments of 100 milliseconds.
- **link-down** Forces the physical link down.
- **manual** Specifies manual mode for a gigabit interface. Setting the gigabit interface to **manual** disables auto negotiation.

Note: You must configure both sides in the same negotiation mode, or the link does not initiate.

phy operating-mode

set interface *interface* phy operating-mode { ... }
unset interface *interface* phy operating-mode { ... }

phy operating-mode

Sets the physical operating mode for the ADSL interface.

- **auto** auto negotiate the operating mode.
- **ansi-dmt** supports data rates up to 8 Mbps downstream and 1 Mbps upstream.
- **glite** supports data rates up to 1.536 Mbps downstream and 512 kbps upstream
- **itu-dmt** supports data rates of 6.144 Mbps downstream and 640 kbps upstream.
- **adsl2** supports data rates up to 1.2 Mbps downstream and 12 Mbps upstream
- **adsl2plus** supports data rates up to 1.2 Mbps downstream and 24 Mbps upstream.

Example: The following command sets the adsl1 interface to use the ADSL2 operating mode:

set interface adsl1 phy operating-mode adsl2

pmtu ipv4

set interface *interface* pmtu ipv4
unset interface *interface* pmtu ipv4

pmtu ipv4	Enables Path MTU (PMTU) for a specified interface. An interface uses the PMTU to determine the size of transmitted packets for each destination host. When the interface attempts to transmit a packet to a destination host through a router that has a link MTU smaller than the packet size, the router returns a PTB (Packet Too Big) error message. When the interface receives this message, it lowers the PMTU setting to the MTU of the router. It then resumes transmission, sizing the outgoing packets according to the new PMTU value. After lowering the PMTU, the interface incrementally resets the PMTU back toward the original MTU value. This allows the security device to increase packet sizes, in case the path changes or more bandwidth becomes available through a midstream router.

port

set interface *interface* port *port_num*

port	Sets the port to be bound to a bridge group. This command allows Ethernet and wireless interfaces to have the same IP address.

Example: The following command sets port ethernet0/2 to be bound to the bridge group 1 (bgroup1) interface:

set interface bgroup1 port ethernet0/2

ppp

get interface *interface* ppp

ppp	Shows the statistics and configuration information for an interface configured for Point-to-Point Protocol (PPP) or Multilink PPP. The interface can be a WAN interface, a bundle interface, or a WAN interface that is a member of a MLPPP bundle interface.

ppp profile

set interface *interface* ppp profile *profile*

ppp profile	Binds the Point-to-Point Protocol (PPP) profile to the specified interface. You configure PPP profiles with the **set ppp profile** command.

primary-number

set interface *interface* primary-number *string*
unset interface *interface* primary-number *string*

primary-number *string*	Specifies the remote destination to call. If the primary number is not connected, **alternative-number** is used. **primary–number** is a string from 1 to 15 characters.

priority

set interface *interface* modem isp *name_str* priority *number*

priority	Specifies the priority of this ISP for dial-up backup, relative to other ISPs that may be configured. A value of 1 is the highest priority. *number* can be 0 or 1-4.

Example: The following command configures the ISP *isp1* as the highest priority for dial-up backup:

set interface serial1/0 modem isp isp1 priority 1

protocol

set interface *interface* protocol igmp host
set interface *interface* protocol igmp router
set interface *interface* protocol igmp { host { ... } | router { ... } }
set interface *interface* protocol ospf
set interface *interface* protocol ospf { ... }
set interface *interface* protocol irdp { ... }
set interface *interface* protocol pim
set interface *interface* protocol pim { ... }
set interface *interface* protocol rip
set interface *interface* protocol rip { ... }

unset interface *interface* protocol bgp
unset interface *interface* protocol bgp { ... }
unset interface *interface* protocol ospf
unset interface *interface* protocol ospf { ... }
unset interface *interface* protocol rip
unset interface *interface* protocol rip { ... }
unset interface *interface* protocol igmp
unset interface *interface* protocol igmp { ... }
unset interface interface protocol irdp { ... }
unset interface *interface* protocol pim
unset interface *interface* protocol pim { ... }
unset interface *interface* protocol irdp { ... }

protocol rip

Sets, unsets, or displays the current RIP settings for the interface.

Note: RIP is *not* supported over unnumbered tunnel interfaces. All interfaces that use RIP protocol must be numbered. Any attempt to configure and run an unnumbered interface using RIP may lead to unpredictable routing failure.

- **authentication** { **password** *pswd_str* | **md5** *key_str* **key-id** *id_num* } Specifies the authentication method used to verify RIP 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.
- **demand-circuit** (For tunnel interfaces only) Enables the demand circuit feature (RFC 2091) on the specified interface.
- **enable** Enables RIP on the specified interface.
- **metric** *number* Configures the RIP metric for the specified interface. The default metric is 1.
- **neighbor** *ip_addr* Configures a static RIP neighbor on the specified interface. This can be used when configuring point-to-multipoint RIP interfaces.
- **passive-mode** Specifies that the interface is to receive but not transmit RIP packets.

- **receive-version v1** | **v1v2** | **v2** Specifies the RIP protocol version for updates that the specified interface receives. The default version is the version that is configured for the virtual router.
- **route-map** *name_str* Specifies the route-map on which to filter incoming routes (routes learned by RIP) or outgoing routes (routes advertised by RIP).
  - **in** Specifies the route map is to be used for incoming routes.
  - **out** Specifies the route map is to be used for outgoing routes.
- **send-version v1** | **v1v2** | **v2** Specifies the RIP protocol version for updates that the specified interface sends. The default version is the version that is configured for the virtual router.
- **split-horizon** Enables the split-horizon function on the specified interface. If **split-horizon** is enabled, RIP 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 advertises routes learned from a neighbor as they exist in the RIP database. By default, **split-horizon** is enabled.

  When you enable the poison-reverse switch, RIP 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 advertises routes learned from a neighbor back to the same neighbor with the correct metric.
- **summary-enable** Enables route summarization in routing updates sent on the specified interface. You configure RIP summary routes at the virtual router level.

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.
- **demand-circuit** (For tunnel interfaces only) Enables the demand circuit feature (RFC 1793) on the specified interface.
- **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.

- **ignore-mtu** Specifies that any mismatches in maximum transmission unit (MTU) values between the local and remote interfaces that are found during OSPF database negotiations are ignored. This option should only be used when the MTU on the local interface is lower than the MTU on the remote interface.
- **link-type** Configures the interface link type. By default, an Ethernet interface 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 desigtnated router for the network.
  - **p2p** Configures the interface as a point-to-point link.
  - **p2mp** (For tunnel interfaces only) Configures the interface as a point-to-multipoint link.
- **neighbor-list** *number* Specifies the number of an access list 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.
- **reduce-flooding** Specifies that periodic LSA updates are not flooded on the specified interface. Other OSPF routers in the area must support the demand circuit feature.
- **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.

protocol igmp

Sets, unsets, or displays the current IGMP settings for the interface.

- **accept groups** Specifies the access list that identifies the multicast groups the hosts on the specified interface can join. Enter this command only if the interface is in router mode.
- **accept hosts** Specifies the access list that identifies from which hosts the interface can receive join and leave messages. After you have set this command, the interface accepts join and leave messages only from the hosts in the access list. Enter this command only if the interface is in router mode.
- **accept routers** Specifies the access list that identifies the routers that are eligible for Querier selection. Only the routers in this list can be elected as Querier. Enter this command only if the interface is in router mode.
- **always** Enables the interface to forward IGMP messages even if it is a non-Querier. Enter this command only if the interface is in Router mode and IGMP proxy is enabled.
- **enable** Enables or disables the IGMP protocol on the interface.
- **host** Creates an IGMP host instance on the specified interface.
- **join-group** Enables the interface to join the specified multicast group. Enter this command only if the interface is in router mode.

- **last-member-query-interval** Sets the interval (in seconds) the Querier waits for a response to a group-specific query before it stops sending multicast traffic for that particular group on the specified interface (range 1-25 inclusive). Enter this command if the interface is in router mode and it is running IGMP version 2.
- **leave-interval** Sets the interval (in seconds) between group specific-queries (range 1 - 255 inclusive). Enter this command if the interface is in router mode.
- **no-check-router-alert** IGMP packets contain a router-alert IP option. By default, an IGMP-enabled device checks IGMP packets for this option and drops packets that do not have this option. Enter this command to accept all IGMP packets without checking for the router-alert option.
- **no-check-subnet** By default, an IGMP interface accepts IGMP packets from its own subnet only. Enter this command to allow the interface to accept IGMP packets (queries, membership reports, and leave messages) from any subnet.
- **proxy** When the interface is in router mode, enables IGMP proxy mode.
- **query-interval** Specifies the interval (in seconds) between General Queries (range 1 to 255, inclusive). Enter this command if the interface is set to router mode and it is the Querier for a multicast group.
- **query-max-response-time** Sets the maximum number of seconds that elapses between the time a Querier sends a General Query and the time a host responds to it (range1 to 25, inclusive). Enter this command if the interface is in router mode.
- **router** Sets the specified interface to router mode.
- **static-group** Manually adds the multicast group to the specified interface. Enter this command only if the interface is in router mode.
- **version** Specifies the IGMP version. When an interface is in host mode, the device automatically sets the IGMP version. When an interface is in router mode, it runs IGMP version 2 by default. Enter this command to change the IGMP version of a router interface. security devices support IGMP versions 1, 2, and 3.

protocol irdp

Sets or unsets the current ICMP Router Discovery Protocol (IRDP) settings for an interface. **Note:** This feature is available only on certain platforms.

- *ip_addr* **{ advertise** | **preference** *number* **}**
  - **advertise** indicates that you want to advertise one of the interface's IP addresses to the network.
  - **preference** indicates the preference status of this device. The value range is -1 to 2147483647. Higher numbers have greater preference.
- **broadcast-address** enables sending of broadcast advertisements. The default address is 224.0.0.1 (all hosts on the network).
- **enable** enables or disables IRDP on the interface. IRDP is disabled by default. Enabling this feature initiates an immediate advertisement to the network. Disabling this feature causes all IRDP-related memory for this interface to be removed. To disable this feature, use the **unset interface** *interface_name* **protocol irdp enable** command.
- **init-adv-interval** *seconds* is the number of seconds during the IRDP startup period allocated for advertisement. The valid value range is 1 through 32 seconds. By default, this period is 16 seconds.

- **init-adv-packet** *seconds* By default, the device sends three advertisement packets during the specified startup period (init-adv-interval). Use this command to change this setting to a number from 1 through 5.
- **lifetime** *seconds* is the lifetime of the advertisement. By default, the lifetime value is three times the max-adv-interval value. The valid value range is the maximum advertisement interval (4 through 1800 seconds) through 9000 seconds.
- **max-adv-interval** *upper_limit* configures the upper limit in seconds. When you change this value, the min-adv-interval and lifetime automatically update to reflect the new upper limit. The default value is 600 seconds. The upper limit can be from 4 through 1800 seconds.
- **min-adv-interval** *lower_limit* is the lower limit of the advertisement period, which is 75 percent of the max-adv-interval value. You can change this value to a number between 3 and the max-adv-interval value. When you change the max-adv-interval value, the min-adv-interval value is automatically calculated.
- **response-delay** *seconds* By default, the device waits 0 to 2 seconds before responding to a client-solicitation request. You can change the response delay setting to no delay (0 seconds) to up to a four-second delay.

protocol pim

Sets, unsets, or displays the current PIM settings for the interface.

- **boot-strap border** Configures the interface as a border for bootstrap (BSR) messages. The interface receives and processes BSR messages, but does not forward these messages to other interfaces even if there is a multicast group policy that allows BSR messages between zones.
- **dr-priority** Configures the priority of the interface during the designated router election.
- **enable** Enables or disables the PIM-SM protocol on the interface.
- **hello-interval** Specifies the interval (expressed in seconds) at which the interface sends hello messages to its neighbors.
- **join-prune-interval** Sets the interval (expressed in seconds) at which the interface sends join-prune messages.
- **neighbor-policy** Identifies the access list that allows or disallows certain neighbor adjacencies.

proxy dns

set interface *interface* proxy dns

proxy dns	Directs the device to use proxy DNS feature. The proxy DNS feature provides a transparent mechanism that allows clients to make split DNS queries. Using this technique, the proxy selectively redirects the DNS queries to specific DNS servers, according to partial or complete domain names. 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 acme.com) could go to the corporate DNS server exclusively, while all others go to the ISP DNS server, thus reducing the load on the corporate server. In addition, this can prevent corporate domain information from leaking into the internet. ■ DNS proxy allows you to transmit selected DNS queries through a tunnel interface, thus preventing malicious users from learning about internal network configuration. For example, DNS queries bound for the corporate server can pass through a tunnel interface, and use security features such as authentication, encryption, and anti-replay.

pvc

set interface *interface* pvc *pvc_num* [mux { vc | llc }] [qos { ... } [protocol { routed | bridged }] zone *zone_name*
unset interface *interface* pvc *pvc_num*

pvc	Specifies the VPI and VCI numbers for an ADSL interface. Valid VPI range is 0-255, default is 8. Valid VCI range is 32-65535, default is 35.
mux	Sets the encapsulating method for carrying network traffic for an ADSL interface. The default mux is LLC.
qos	Sets the ATM QoS type.The qos keyword options are as follows: ■ **cbr** specifies the CBR service class ■ **ubr** specifies the UBR service class ■ **vbr-nrt** specifies the VBR-NRT service class
protocol	Sets the protocol type for an ADSL interface. The default protocol is bridged.

Example: The following command sets the adsl1 interface to have a pvc of 1/35, the mux as vc, and binds the interface to the DMZ security zone:

set interface adsl1 pvc 1/35 mux vc zone dmz

retry

set interface *interface* modem retry *number*
set interface *interface* retry *number*
unset interface *interface* modem retry *number*
uset interface *interface* retry *number*

retry	Specifies the number of times ScreenOS dials the primary number, and then the alternative-number, if the line is busy or there is no answer from the ISP. The default is 3 times. The range is 0-10 times.

Example1: The following command sets the number of dial-up retries to 4:

set interface serial1/0 modem retry 4

Example2: The following command sets the number of dial-up retries to 4 for the basic rate interface (ISDN):

set interface bri1/0 retry 4

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.

serial-options

set interface *interface* serial-options ...

serial-options Specifies options for a serial interface. You can specify the following:

- **clock-rate** Sets the clock rate for the interface, in Kilohertz (KHz) or Megahertz (MHz), for EIA-530 and V.35 interfaces (for X.21 interfaces, you must specify **loop** for the **clocking-mode** option). The default is 8.0 MHz. You can specify one of the following options:
  - **1.2khz**
  - **2.4khz**
  - **9.6khz**
  - **19.2khz**
  - **38.4khz**
  - **56.0khz**
  - **64.0khz**
  - **72.0khz**
  - **125.0khz**
  - **148.0khz**
  - **250.0khz**
  - **500.0khz**
  - **800.0khz**
  - **1.0mhz**
  - **1.3mhz**
  - **2.0mhz**
  - **4.0mhz**
  - **8.0mhz**
- **clocking-mode** Specifies the clock source to determine the timing on serial interfaces. You can specify one of the following:
  - **dce** Uses a transmit clock generated by the data circuit-terminating equipment (DCE) for the SSG device's DTE. When the device is functioning as a DTE, you must use this clocking mode for all interfaces except X.21 serial interfaces.
  - **internal** Uses the SSG device's internal clock. When the device is functioning as a DCE, we recommend that you use this clocking mode for all interfaces. You can configure the speed of the clock with the **clock-rate** option.
  - **loop** Uses the DCE's or DTE's receive clock. For X.21 serial interfaces, you must use this clocking mode. This is the default.

- **dte-options**. Sets data terminal equipment (DTE) options/control leads. You can specify the following:
  - **cts** Specifies the from-DCE clear to send (CTS) signal handling for EIA-530 and V.35 interfaces. You can specify one of the following:
    - **ignore** Ignores CTS signal.
    - **normal** Normal CTS signal, as defined by TIA/EIA Standard 530. This is the default.
    - **require** The from-DCE CTS signal must be asserted.
  - **dcd** Specifies the from-DCE data carrier detect (DCD) signal handling for EIA-530 and V.35 interfaces. You can specify one of the following:
    - **ignore** Ignores DCD signal.
    - **normal** Normal DCD signal, as defined by TIA/EIA Standard 530. This is the default.
    - **require** The from-DCE DCD signal must be asserted.
  - **dsr** Specifies the from-DCE data set ready (DSR) signal handling for EAI-530 and V.35 interfaces. You can specify one of the following:
    - **ignore** Ignores DSR signal.
    - **normal** Normal DSR signal, as defined by TIA/EIA Standard 530. This is the default.
    - **require** The from-DCE DSR signal must be asserted.
  - **dtr** Specifies data transmit ready (DTR) signal handling for EIA-530 and V.35 interfaces. You can specify one of the following:
    - **assert** Asserts the DTR signal.
    - **auto-synchronize** Normal DTR signal, with automatic resynchronization.
    - **de-assert** Deasserts the DTR signal.
    - **normal** Normal DTR signal, as defined by TIA/EIA Standard 530. This is the default.
  - **ignore-all** Specifies that all control leads are ignored. By default, this is not set.
  - **rts** Specifies the to-DCE request to send (RTS) signal handling for EIA-530 and V.35 interfaces. You can specify one of the following:
    - **assert** Assertd RTS signal.
    - **de-assert** Deassertd RTS signal.
    - **normal** Normal RTS signal, as defined by TIA/EIA Standard 530. This is the default.
  - **tm** Specifies the test mode (TM) signal for EIA-530 interfaces. You can specify one of the following:
    - **ignore** Ignored TM signal.
    - **normal** Normal TM signal. This is the default.
    - **require** The from-DCE TM signal must be asserted.
- **encoding** Sets line encoding. You can specify one of the following:
  - **nrz** Nonreturn-to-zero. This is the default.
  - **nrzi** Nonreturn-to-zero-inverted.

- **loopback** Sets loopback mode. By default, no loopback mode is specified. You can specify one of the following:
  - **dce-local** DCE local loopback (DTE mode only).
  - **local** Local loopback.
  - **remote** Remote/line interface unit (LIU) loopback.
- **transmit-clock** Sets the transmit-clock phase. By default, this is not set. You can specify the following:
  - **invert** Shift clock phase 180 degrees.

settings

set interface *interface* modem settings *name_str* active | init-strings *string*
unset interface *interface* modem settings *name_str*
get interface *interface* modem settings

settings	Configures settings for the specified modem or ISP.

Example: The following command activates settings for the modem *usr14400*:

set interface serial1/0 modem settings usr14400 active

short-sequence

set interface *bundle* short-sequence

short-sequence	(For MLPPP bundle interfaces only) Specifies a sequence-header format of 12 bits. The default is 24 bits.

shutdown

set interface *interface* shutdown
unset interface *interface* shutdown

shutdown	Disables a wireless interface. Also disables 802.1X on the interface.

speed

set interface *interface* modem speed *number*
unset interface *interface* modem speed

speed	Specifies the maximum baud rate for the serial link between the device and the modem. The baud rate can be 9600, 19200, 38400, 57600, or 115200 bps. The default is 115200 bps.

Example: The following command sets a maximum baud rate of 56Kbps for the serial link:

set interface serial1/0 modem speed 57600

t1-options

set interface *interface* t1-options ...

t1-options

Specifies options for a T1 interface. You can specify the following:

- **bert-algorithm** Sets the bit error rate testing (BERT) algorithm for the interface. The algorithm is the pattern to send in the bitstream. You can specify one of the following options:
  - **all-ones-repeating** Repeating one bits.
  - **all-zeros-repeating** Repeating zero bits.
  - **alternating-double-ones-zeros** Alternating pairs of ones and zeroes.
  - **alternating-ones-zeros** Alternating ones and zeroes.
  - **pseudo-2e10** Pattern is 2^10-1.
  - **pseudo-2e11-o152** Pattern is 2^11-1 (per O.152 standard).
  - **pseudo-2e15-o151** Pattern is 2^15-1 (per O.152 standard). This is the default.
  - **pseudo-2e17** Pattern is 2^17-1.
  - **pseudo-2e18** Pattern is 2^18-1.
  - **pseudo-2e20-o151** Pattern is 2^20-1 (per O.151 standard).
  - **pseudo-2e20-o153** Pattern is 2^20-1 (per O.153 standard).
  - **pseudo-2e21** Pattern is 2^21-1.
  - **pseudo-2e22** Pattern is 2^22-1.
  - **pseudo-2e23-o151** Pattern is 2^23 (per O.151 standard).
  - **pseudo-2e25** Pattern is 2^25-1.
  - **pseudo-2e28** Pattern is 2^28-1.
  - **pseudo-2e29** Pattern is 2^29-1.
  - **pseudo-2e3** Pattern is 2^3-1.
  - **pseudo-2e31** Pattern is 2^31-1.
  - **pseudo-2e32** Pattern is 2^32-1.
  - **pseudo-2e4** Pattern is 2^4-1.
  - **pseudo-2e5** Pattern is 2^5-1.
  - **pseudo-2e6** Pattern is 2^6-1.
  - **pseudo-2e7** Pattern is 2^7-1.
  - **pseudo-2e9-o153** Pattern is 2^9-1 (per O.153 standard).
  - **repeating-1-in-4** One bit in 4 is set.
  - **repeating-1-in-8** One bit in 8 is set.
  - **repeating-3-in-24** Three bits in 24 are set.
- **bert-error-rate** Sets the bit error rate (BER) to use in BERT. This can be an integer from 0 to 7, which corresponds to a BER from 10^{-0} (1 error per bit) to 10^{-7}. The default is 0.
- **bert-period** Sets the length of the BERT, in seconds. The default is 10. Specify a value between 1 and 240 seconds.

- **buildout**. Sets the T1 cable length in feet. You can specify the following:
  - **0-132** 0-40 meters. This is the default.
  - **133-265** 40-81 meters.
  - **266-398** 81-121meters.
  - **399-531** 121-162 meters.
  - **532-655** 162-200 meters.
- **byte-encoding** Sets the byte-encoding method. You can specify one of the following:
  - **nx56** Seven bits per byte.
  - **nx64** Eight bits per byte. This is the default.
- **fcs** Specifies the number of bits in the frame checksum. You can specify one of the following:
  - **16** 16 bits. This is the default.
  - **32** 32 bits.
- **framing** Sets the framing mode for the T1 line. You can specify the following:
  - **esf** Extended superframe. This is the default.
  - **sf** Superframe.
- **idle-cycle-flag** Sets the value to transmit in idle cycles. You can specify one of the following:
  - **flags** Transmit 0x7E in idle cycles. This is the default.
  - **ones** Transmit 0xFF (all ones) in idle cycles.
- **invert-data** Specifies data inversion. Data inversion is normally used only in alternate mark inversion (AMI) mode. By default, this is not set.
- **line-encoding** Specifies the line-encoding method. You can specify one of the following:
  - **ami** Alternate mark inversion.
  - **b8zs** Binary 8 zero substitution. This is the default.
- **loopback** Specifies loopback mode. By default, no loopback mode is set. You can specify one of the following:
  - **local** Local loopback.
  - **payload** Payload loopback.
  - **remote** Remote loopback.
- **remote-loopback-respond** Specifies that the interface responds to loop requests from the remote end. By default, this is not set.
- **start-end-flag** Sets the start and end flags on transmission. You can specify one of the following:
  - **filler** Sends two idle cycles between start/end flags. This is the default.
  - **shared** Shares start/end flags on transmit.
- **timeslots** *timeslots* Specifies the number of time slots allocated to a fractional T1 interface. By default, all time slots are active. Specify values from 1 to 24. Use hyphens to specify a range. Use commas (with no spaces before or after) to separate individual time slots or ranges. For example, you can specify the following: 1-3,4,9,22-24.

t3-options

set interface *interface* t3-options ...

t3-options

Specifies options for a T3 interface. You can specify the following:

- **bert-algorithm** Sets the bit error rate testing (BERT) algorithm for the interface. The algorithm is the pattern to send in the bitstream. You can specify one of the following options:
  - **all-ones-repeating** Repeating one bits.
  - **all-zeros-repeating** Repeating zero bits.
  - **alternating-double-ones-zeros** Alternating pairs of ones and zeroes.
  - **alternating-ones-zeros** Alternating ones and zeroes.
  - **pseudo-2e10** Pattern is 2^10-1.
  - **pseudo-2e11-o152** Pattern is 2^11-1 (per O.152 standard).
  - **pseudo-2e15-o151** Pattern is 2^15-1 (per O.152 standard). This is the default.
  - **pseudo-2e17** Pattern is 2^17-1.
  - **pseudo-2e18** Pattern is 2^18-1.
  - **pseudo-2e20-o151** Pattern is 2^20-1 (per O.151 standard).
  - **pseudo-2e20-o153** Pattern is 2^20-1 (per O.153 standard).
  - **pseudo-2e21** Pattern is 2^21-1.
  - **pseudo-2e22** Pattern is 2^22-1.
  - **pseudo-2e23-o151** Pattern is 2^23 (per O.151 standard).
  - **pseudo-2e25** Pattern is 2^25-1.
  - **pseudo-2e28** Pattern is 2^28-1.
  - **pseudo-2e29** Pattern is 2^29-1.
  - **pseudo-2e3** Pattern is 2^3-1.
  - **pseudo-2e31** Pattern is 2^31-1.
  - **pseudo-2e32** Pattern is 2^32-1.
  - **pseudo-2e4** Pattern is 2^4-1.
  - **pseudo-2e5** Pattern is 2^5-1.
  - **pseudo-2e6** Pattern is 2^6-1.
  - **pseudo-2e7** Pattern is 2^7-1.
  - **pseudo-2e9-o153** Pattern is 2^9-1 (per O.153 standard).
  - **repeating-1-in-4** One bit in 4 is set.
  - **repeating-1-in-8** One bit in 8 is set.
  - **repeating-3-in-24** Three bits in 24 are set.
- **bert-error-rate** Sets the bit error rate (BER) to use in BERT. This can be an integer from 0 to 7, which corresponds to a BER from 10^{-0} (1 error per bit) to 10^{-7}. The default is 0.
- **bert-period** Sets the length of the BERT in seconds. The default is 10. Specify a value between 1 and 240 seconds.

- **cbit-parity** Disables or enables C-bit parity mode, which controls the type of framing that is present on the transmitted T3 signal. By default, C-bit parity mode is enabled. When C-bit parity mode is enabled, the C-bit positions are used for the FEBE, FEAC, terminal-data-link, path-parity, and mode-indicator bits, as defined in ANSI T1.107a-1989. When C-bit parity mode is disabled, the basic T3 framing mode (M13) is used.

- **compatibility-mode** Sets the T3 interface to be compatible with the channel service unit (CSU) at the remote end of the line. By default, no compatibility mode is set. You can specify one of the following:
  - **adtran subrate** For intelligent-queuing (IQ) channels only. Sets the interface to be compatible with Adtran Channel Service Units (CSUs). Specify a value between 1 and 588.

- **digital-link subrate** Sets the interface to be compatible with Digital Link CSUs. Specify one of the following bits-per-second values:
  - **301Kb**
  - **601Kb**
  - **902Kb**
  - **1.2Mb**
  - **1.5Mb**
  - **1.8Mb**
  - **2.1Mb**
  - **2.4Mb**
  - **2.7Mb**
  - **3.0Mb**
  - **3.3Mb**
  - **3.6Mb**
  - **3.9Mb**
  - **4.2Mb**
  - **4.5Mb**
  - **4.8Mb**
  - **5.1Mb**
  - **5.4Mb**
  - **5.7Mb**
  - **6.0Mb**
  - **6.3Mb**
  - **6.6Mb**
  - **6.9Mb**
  - **7.2Mb**
  - **7.5Mb**
  - **7.8Mb**
  - **8.1Mb**
  - **8.4Mb**
  - **8.7Mb**
  - **9.0Mb**
  - **9.3Mb**
  - **9.6Mb**
  - **9.9Mb**
  - **10.2Mb**
  - **10.5Mb**
  - **10.8Mb**
  - **11.1Mb**
  - **11.4Mb**
  - **11.7Mb**
  - **12.0Mb**
  - **12.3Mb**
  - **12.6Mb**
  - **12.9Mb**
  - **13.2Mb**
  - **13.5Mb**
  - **13.8Mb**
  - **14.1Mb**
  - **14.4Mb**
  - **14.7Mb**
  - **15.0Mb**
  - **15.3Mb**
  - **15.6Mb**
  - **15.9Mb**
  - **16.2Mb**
  - **16.5Mb**
  - **16.8Mb**
  - **17.1Mb**
  - **17.4Mb**
  - **17.7Mb**
  - **18.0Mb**
  - **18.3Mb**
  - **18.6Mb**
  - **18.9Mb**
  - **19.2Mb**
  - **19.5Mb**
  - **19.8Mb**
  - **20.1Mb**
  - **20.5Mb**
  - **20.8Mb**
  - **21.1Mb**
  - **21.4Mb**
  - **21.7Mb**
  - **22.0Mb**
  - **22.3Mb**
  - **22.6Mb**
  - **22.9Mb**
  - **23.2Mb**
  - **23.5Mb**
  - **23.8Mb**
  - **24.1Mb**
  - **24.4Mb**
  - **24.7Mb**
  - **25.0Mb**
  - **25.3Mb**
  - **25.6Mb**
  - **25.9Mb**
  - **26.2Mb**
  - **26.5Mb**
  - **26.8Mb**
  - **27.1Mb**
  - **27.4Mb**
  - **27.7Mb**
  - **28.0Mb**
  - **28.3Mb**
  - **28.6Mb**
  - **28.9Mb**
  - **29.2Mb**
  - **29.5Mb**
  - **29.8Mb**
  - **30.1Mb**
  - **30.4Mb**
  - **30.7Mb**
  - **31.0Mb**
  - **31.3Mb**
  - **31.6Mb**
  - **31.9Mb**
  - **32.2Mb**
  - **32.5Mb**
  - **32.8Mb**
  - **33.1Mb**
  - **33.4Mb**
  - **33.7Mb**
  - **34.0Mb**
  - **34.3Mb**
  - **34.6Mb**
  - **34.9Mb**
  - **35.2Mb**
  - **35.5Mb**
  - **35.8Mb**
  - **36.1Mb**
  - **36.4Mb**
  - **36.7Mb**
  - **37.0Mb**
  - **37.3Mb**
  - **37.6Mb**
  - **37.9Mb**
  - **38.2Mb**
  - **38.5Mb**
  - **38.8Mb**
  - **39.1Mb**
  - **39.4Mb**
  - **39.7Mb**
  - **40.0Mb**
  - **40.3Mb**
  - **40.6Mb**
  - **40.9Mb**
  - **41.2Mb**
  - **41.5Mb**
  - **41.8Mb**
  - **42.1Mb**
  - **42.4Mb**
  - **42.7Mb**
  - **43.0Mb**
  - **43.3Mb**
  - **43.6Mb**
  - **43.9Mb**
  - **44.2Mb**
- **kentrox subrate** For IQ channels only. Sets the interface to be compatible with Kentrox CSUs. Specify a value between 1 and 69.
- **larscom subrate** For IQ channels only. Sets the interface to be compatible with Larscom CSUs. Specify a value between 1 and 14.

- **verilink subrate** For IQ channels only. Sets the interface to be compatible with Verilink CSUs. Specify a value between 1 and 28.
- **fcs** Specifies the number of bits in the frame checksum. The checksum must be the same on both ends of the link. You can specify one of the following:
  - **16** 16 bits. This is the default.
  - **32** 32 bits.
- **feac-loop-respond** Sets the interface to respond to far-end alarm and control (FEAC) loop requests. By default, this is not set.
- **idle-cycle-flag** Sets the value to transmit in idle cycles. You can specify one of the following:
  - **flags** Transmit 0x7E in idle cycles. This is the default.
  - **ones** Transmit 0xFF (all ones) in idle cycles.
- **long-buildout** Specifies a long cable length (longer than 225 feet or 68.6 meters) for copper-cable-based T3 interfaces. By default, this is not set.
- **loopback** Specifies loopback mode. By default, no loopback mode is set. You can specify one of the following:
  - **local** Local loopback.
  - **payload** Payload loopback.
  - **remote** Remote loopback.
- **payload-scrambler** Enables High-Level Data Link Control (HDLC) payload scrambling on the interface. This type of scrambling provides better link stability, but both sides of the connection must either use or not use scrambling. By default, this is not set.
- **start-end-flag** Sets the start and end flags on transmission. You can specify one of the following:
  - **filler** Sends two idle cycles between start/end flags. This is the default.
  - **shared** Shares start/end flags on transmit.

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 information on interface names, see “Interface Names” on page A-I.

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 { ... }

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 IP tracking on the interface. If the weighted sum of all tracked IP failures on the interface is equal to or greater than the threshold, IP tracking on the interface is considered to be failed and the routes associated with the interface are deactivated on the security device. On some security devices, failover to the backup interface 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.

Example 1: 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

Example 2: 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 [demand-circuit] | rip [demand-circuit] } { ... } }
set interface tunnel.*n* tunnel encap gre [key]
set interface tunnel.*n* tunnel keep-alive interval *number*
set interface tunnel.*n* tunnel keep-alive threshold *number*
set interface tunnel.*n* tunnel local-if *interface* dst-ip *ip_addr*
unset interface tunnel.*n* tunnel
unset interface tunnel.*n* tunnel [keep-alive]
unset interface tunnel.*n*

tunnel.*n*	Specifies a tunnel interface. The *n* parameter is an ID number that identifies the tunnel interface.
tunnel	Specifies parameters for the tunnel interface.
encap gre	Specifies that all traffic in the tunnel is encapsulated using the GRE (Generic Routing Encapsulation) protocol.
keep-alive	The tunnel interface sends keep-alive messages to monitor the status of the connection. You can specify the interval (in seconds) between keep-alive messages, and the number of times the local tunnel interface sends keep-live messages without receiving a reply before it terminates the connection.
local-if	Specifies the local interface and the destination IP address of a GRE tunnel.

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

protocol

set interface tunnel.*n* protocol { bgp | ospf [demand-circuit] | rip [demand-circuit] } { ... } }
unset set interface tunnel.*n* protocol { bgp | ospf [demand-circuit] | rip [demand-circuit] } { ... } }

protocol	Specifies the routing protocol that the device uses on a specified tunnel interface. security devices support BGP, OSPF, RIP, IGMP, and PIM. These commands set or unset protocol parameters.

Example: The following command enables the RIP-specific route summary feature for the *tunnel.1* interface:

set interface tunnel.1 protocol rip summary-enable

vip

set interface *interface* vip *ip_addr*
[+] *port_num* [*name_str ip_addr* [manual]]

vip	Defines a Virtual IP (VIP) address (ip_addr) for 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.

- Drop the frames because they have tags.
- Ignore the tags and forward 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 [ssl-only]

webauth	Enables WebAuth user authentication. Enabling the **ssl-only** switch allows only SSL-based (HTTPS) 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 user name and password. After successfully entering the user name 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.

wlan

set interface *interface* wlan { 0 | 1 | both }
unset interface interface wlan

wlan	Specifies the wireless operational mode for a wireless interface. ■ **0** Specifies that 802.11b and 802.11g are available. ■ **1** Specifies that 802.11a is available. ■ **both** Specifies that 802.11a, 802.11b, and 802.11g are available.

zone

set interface *interface* zone *zone*
unset interface *interface* zone

zone	Binds the interface to a security zone.

Example: To bind interface *ethernet2/2* to the Trust zone:

set interface ethernet2/2 zone trust

ip

Use the **ip** commands to set or display Internet Protocol (IP) parameters for communication with a Trivial File Transfer Protocol (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. The default is 10.

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. The default is 2 seconds.

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 Internet Protocol (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 695.

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.It has to be a shared zone in a shared virtual router. A virtual system (vsys) must also be enabled. This command is only available in root vsys. **ip** *ip_addr* specifies a specific address in a specific zone.

ippool

Use the **ippool** commands to associate the name of an Internet Protocol (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

clear

clear ippool *name_str* [*ip_addr1 ip_addr2*]

get

get ippool [*name_str*]

set

set ippool *name_str ip_addr1 ip_addr2*
set ippool *name_str ip_addr3 ip_addr4*

Keywords and Variables

Variable Parameters

clear ippool *name_str* [*ip_addr1 ip_addr2*]
get ippool *name_str*
set ippool *name_str ip_addr1 ip_addr2*
set ippool *name_str ip_addr3 ip_addr4*
unset ippool *name_str*

name_str	Defines the name of the IP pool.
ip_addr1 *ip_addr2*	Starting and ending IP addresses in the IP pool.
ip_addr3 *ip_addr4*	A second set of starting and ending IP addresses in the same 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

ipsec

Use the **ipsec** commands to view Internet Protocol Security (IPSec) session information.

Syntax

get

```
get ipsec access-session
    {
    all | id | ike-cfg-ipv4-address ip_addr | ipv4-address ip_addr | ipv6-address ip_addr |
    status | xauth-user-name string
    }
```

set

```
set ipsec access-session
    [
    dead-p2-sa-timeout number|
    enable |
    info-exch-connected |
    log-error |
    lower-threshold number |
    maximum number |
    upper-threshold number
    ]
```

Keywords and Variables

ipsec

get ipsec access-session { ... }

ipsec access-session	■ **dead-p2-sa-timeout** *number* defines the timeout value in seconds for a dead p2 sa. ■ **enable** enables the ipsec access-session feature or restores the default values if the feature was previously enabled. ■ **info-exch-connected** sends a "connected" notification by information exchange. ■ **log-error** enables the IAS error log. ■ **lower-threshold** *number* defines the minimum number of ipsec access sessions. ■ **maximum** *number* defines the maximum allowable number of ipsec access sessions. ■ **upper-threshold** *number* defines the upper limit of the ipsec access session range.

irdp

Use the **irdp** commands to view a configured ICMP Router Discovery Protocol (IRDP) instance for an interface of your security device.

NOTE: This protocol is not available on all platforms. Refer to your data sheet for a list of features available for your particular platform.

To configure an IRDP instance, see "interface" on page 249 for command syntax and explanations for using the commands.

Syntax

get

get irdp [*interface*]

Keywords and Variables

Variable Parameter

get irdp *interface*

interface	The name of the interface. For more information, see "Interface Names" on page A-I.

l2tp

Use the **l2tp** commands to configure or remove Layer 2 Tunneling Protocol (L2TP) tunnels and L2TP settings from the security device.

L2TP is an extension to Point-to-Point Protocol (PPP) that allows Internet Service Providers (ISPs) to operate virtual private networks (VPNs). L2TP allows dial-up users to make virtual 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

Variable Parameter

get l2tp *tunn_str*
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 host name, 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* specifies the name of the authentication server.
- **dns1** *ip_addr* specifies the IP address of the primary DNS server.
- **dns2** *ip_addr* specifies the IP address of the secondary DNS server.
- **ippool** *string* specifies 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* specifes the port number of the default L2TP server. The number can be between 1024 and 65,535.
- **wins1** *ip_addr* specifies the IP address of the primary WINS server.
- **wins2** *ip_addr* specifies 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 host name (*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*.

lcd

Use the **lcd** commands to activate or inactivate the LCD on the front panel of a security device or to display the current **lcd** setting.

Syntax

get

```
get lcd
```

set

```
set lcd
    {
    display |
    key-in
    }
```

Keywords and Variables

display

```
set lcd display
unset lcd display
```

display	Turns the LCD off or on and locks the control keys.

key-in

```
set lcd key-in
unset lcd key-in
```

key-in	Locks and unlocks the control keys but does not affect the LCD display.

led

When either an event alarm or a firewall attack occurs, the LED glows red to signal the attack. Use the **clear led** commands to return an ALARM or firewall (FW) LED to green after such an attack occurs.

Syntax

```
clear [ cluster ] led { alarm | firewall }
```

Keywords and Variables

alarm

```
clear [ cluster ] led alarm
```

alarm	Specifies the ALARM LED.

cluster

```
clear cluster led alarm
clear cluster led firewall
```

cluster	Propagates the **clear** operation to all other devices in a NetScreen Redundancy Protocol (NSRP) cluster.

firewall

```
clear [ cluster ] led firewall
```

firewall	Specifies the FW LED.

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
- NetScreen Redundancy Protocol (NSRP)
- Virtual systems
- Virtual private networks (VPNs)
- Zones
- Virtual routers
- High availability (HA)

NOTE: Not all keys are available for all products.

Syntax

exec

exec license-key
{
capacity *key_str* |
delete *key_str* |
key_str |
nsrp *key_str* |
update [trial] |
virtualization *key_str* |
vpn *key_str* |
vrouter *key_str* |
vsys *key_str* |
zone *key_str*
}

get

get license-key

set

set license-key update-url *url_str*

Keywords and Variables

Variable Parameters

exec license-key *key_str*
set license-key update-url *url_str*

key_str	The provided license key string.
url_str	The URL of the license key server.

capacity

exec license-key capacity *key_str*

capacity	Allows you to expand the user capacity of the security device with your given license-key (*key_str*).

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 [trial]

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**. Use the trial command to try service temporarily.

- **trial** Updates the trial license-key.

For more information, refer to the *ScreenOS Concepts & Examples 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.

virtualization

exec license-key virtualization *key_str*

virtualization	Specifies a virtualization license key (*key_str*). Virtualization key is used to control VLAN support on some devices. Security devices with VSYS support by default have VLAN support.

vpn

exec license-key vpn *key_str*

vpn	Specifies a Virtual Private Network (VPN) license key (*key_str*).

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 to perform the following actions:

- 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
- Mitigate message loss caused by 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 [-port_num ] ] ]
            ]
                  [ dst-ip ip_addr [ -ip_addr ] [ dst-netmask mask ] ]
                    [ dst-port port_num [-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*
traffic detail level { 0 | 1 }
}

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

detail

set log traffic detail { 0 | 1 }

detail	By default the device reports the reason for session close. If you do not want to view the reason for close, enter zero (0). To return the device to default behavior, enter 1.

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. See also “traffic detail level” for more options.

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* [-*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, note that because **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 *time* end-time *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 [...]
set log traffic detail level { 0 | 1 }
unset log traffic detail level

traffic	Specifies traffic log entries. By default, the security device shows the reason for each log entry. If you do not want to view the reason for each log entry, you can disable this feature by entering the **set log traffic detail level 0** command. To return the security device default behavior, you can enter the **unset log traffic detail level** command.

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 physical security 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

get

get mac [*interface*]

set

set mac *mac_addr interface*

Keywords and Variables

Variable Parameters

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.

mac-learn 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*]

set

set mac-learn

Keywords and Variables

Variable Parameter

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.

match-group

Use the **match-group** commands to configure the security device for setting Policy-Based Routing (PBR).

Syntax

set

set match-group { name *match_group_name* |
match_group_name ext-acl *ext_acl_id* match-entry *group-entry-id* }

Keywords and Variables

match-group

set match-group name *match_group_name*
set match-group *match_group_name* ext-acl *ext_acl_id* match-entry *group-entry-id*
unset match-group *match_group_name* ext-acl *ext_acl_id* match-entry *group-entry-id*

match-group	Specifies the name of a match group. Each match-group name must be a unique alphanumeric string and must be between 1 and 28 characters in length. Once a match-group name is defined, then you can associate or remove an extended access-list. You can use match groups to group multiple extended access-lists. Match group entries are evaluated sequentially by group entry id number. You can combine multiple extended access-lists to a single match group and then assign the single match group to an action group. An action group can have multiple forwarding solutions and an associated lookup sequence number. To configure an action group, refer to the *Concepts & Examples ScreenOS Reference Guide.* To remove an access-list, enter **unset match-group** *match_group_name* **match-entry** *group_entry_id*

memory

Use the **memory** commands to set or display memory-allocation settings.

Syntax

get

```
get memory
    [
      [ di | ipc | kernel | id_num | module { all | id_num } ]
        [ all | bin | error | free | used ]
      [ chunk | pool ]
        [ name [ name_str ] | task [ id_num ] ]
    ]
```

Keywords and Variables

Variable Parameters

get memory *id_num*

id_num	The task ID number.

all

get memory all

all	Displays all memory fragments in the device.

bin

get memory bin

bin	Displays the task memory bin.

chunk

get memory chunk [...]

chunk	Displays the object pool (*name_str*) memory.

di

get memory di [...]

di	Displays the statistics about the memory used by the deep inspection module.

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 the memory used by the inter-process communication (IPC).

kernel

get memory kernel [...]

kernel	Displays memory statistics about the kernel heap.

module

get memory module { ... }

module	Displays all or a single memory module (*id_num*).

pool

get memory pool [...]

pool	Displays pooled memory.

used

get memory used

used	Displays used memory.

mip

Use the **mip** command to show all mapped IPs (MIPs) in a specified virtual system (vsys) or root system.

Syntax

get mip [all]

Keywords and Variables

all

get mip all

all	Displays all MIPs in a specified vsys or root.

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 as a result of 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.

modem

Use the **modem** commands to configure modem and dial-up settings for the serial link.

Syntax

exec

```
exec modem { 0 | 1/0 | 2/0 }
    [
      command string |
      dialup |
      stop |
      connect isp_name_str |
      disconnect
    ]
```

get

```
get modem { 0 | 1/0 | 2/0 }
    [
    config |
    queue { rcv-q | xmt-q } |
    settings |
    state |
    stats
    ]
```

set

```
set modem { 0 | 1/0 | 2/0 }
    {
    idle-time number |
    interval number |
    isp name_str
      {
      account login string password pswd_str |
      primary-number string [ alternative-number string ] |
      priority number
      }
    isp-failover
      {
      holddown number |
      type { route | track-ip | vpn } vrouter vr_name ipaddr/mask
      } |
    retry number |
    settings name_str { active | init-strings string } |
    speed number
    }
```

Keywords and Variables

account login

set modem isp *name_str* account login *string* password *pswd_str*

account login	Specifies the login name (*string*)and account password (*pswd_str)* for the ISP account.

Example: The following command configures the login kgreen and the password bodie45 for the ISP account *isp1*:

set modem isp isp1 account login kgreen password bodie45

active

set modem settings *name_str* active
unset modem settings *name_str*

active	Activates the specified modem settings and deactivates any other configured settings.

Example: The following command activates settings for the modem *usr14400*:

set modem settings usr14400 active

alternative-number

set modem isp *name_str* primary-number *string* alternative-number *string*

alternative-number	Specifies an alternate phone number to access the ISP.

Example: The following command configures primary and alternate phone numbers to access the ISP ‘isp1’:

set modem isp isp1 primary-number 4085551212 alternative-number 4085551313

command

exec modem command *string*

command	Sends Hayes AT commands to the modem.

config

get modem config

config	Displays HDLC/PPP parameters for a current session.

connect

exec modem connect

connect	Connects the device to a specific ISP for testing.

dialup

exec modem dialup

dialup	Enables dialup to start. If the first ISP fails, the device will try other ISPs. Traffic is monitored on the serial interface.

disconnect

exec modem disconnect

disconnect	Disconnets the current connection.

idle-time

set modem idle-time *number*
unset modem idle-time *number*

idle-time	Specifies the number of minutes that elapse with no traffic on the dial-up connection before the security device disconnects the modem. The default is 1 minute. A value of 0 means the modem never disconnects, even if there is no traffic on the dial-up connection.

Example: The following command sets an idle time of 12 minutes:

set modem idle-time 12

init-strings

set modem settings *name_str* init-strings *string*
unset modem settings *name_str*

init-strings	Specifies the initialization string for the specified modem. AT string command that is recognized by the modem.

Example: The following command sets an initialization string for the modem usr14400:

set modem settings usr14400 init-strings
AT&FX4&A3&B1&D2&H1&I0&K1&M4&R2S7=60

interval

set modem interval *number*
unset modem interval *number*

interval	Specifies the seconds (*number*) between dial-up retries. The default is 60 seconds. Range is 3-60 seconds.

Example: The following command sets a dial-up interval of 45 seconds:

set modem interval 45

isp

set modem isp *name_str* { ... }
unset modem isp *name_str*

isp	Specifies the ISP.

Example: The following command configures the login *juniper* and the password *bodie45* for the ISP *isp1*:

set modem isp isp1 account login juniper password bodie45

isp-failover

set modem isp-failover holddown *number*
set modem isp-failover type { route | track-ip | vpn } vrouter *vr_name ipaddr/mask*
unset modem isp-failover holddown
unset modem isp-failover type

isp-failover — Allows you to configure up to four ISPs for failover and dial-up connections. The holddown timer and type arguments can be configured as follows:

- **holddown** *number* specifies the number of seconds to wait before initiating failover. The default value is 30 seconds; however, the valid range is between 1 and 300 seconds. The **unset** command returns the holddown value to the default. Using the **set** command twice overwrites the previous value.
- **type { ... } vrouter** *vr_name ip_addr/mask* specifies a route generated by a dynamic routing protocol, such as OSPF or BGP. The security device monitors the status of the interface in the virtual router. this feature is disabled by default.

primary-number

set modem isp *name_str* primary-number *string*

primary-number	Specifies the primary phone number to access the ISP. If your modem uses tone dial by default, but you want to use pulse dial, precede the phone number with a **P**. If your modem uses pulse dial by default, but you want to use tone dial, precede the phone number with a **T**.

Example: The following command configures the primary phone number to access the ISP isp1 and specifies tone dial:

set modem isp isp1 primary-number T4085551212

priority

set modem isp *name_str* priority *number*

priority	Specifies the priority of this ISP for dial-up backup, relative to other ISPs that may be configured. A value of 1 is the highest priority. The *number* can be 0 or 1-4.

Example: The following command configures the ISP *isp1* as the highest priority for dial-up backup:

set modem isp isp1 priority 1

queue

set modem queue {...}
get modem queue {...}

rcv-q	Displays contents for the HDLC rcv queue. Used for debugging only.
xmt-q	Displays contents for the HDLC xmt queue. Used for debugging only.

Example: The following command displays the content of the HDLC rcv queue:

set modem queue rcv-q

retry

set modem retry *number*
unset modem retry *number*

retry	Specifies the number of times ScreenOS dials the primary number, and then the alternative-number, if the line is busy or there is no answer from the ISP. The default is 3 times. The range is 0-10 times.

Example: The following command sets the number of dial-up retries to 4:

set modem retry 4

settings

set modem settings *name_str* active | init-strings *string*
unset modem settings *name_str*
get modem settings

settings	Configures settings for the specified modem or ISP.

Example: The following command activates settings for the modem *usr14400*:

set modem settings usr14400 active

speed

set modem speed *number*
unset modem speed

speed	Specifies the maximum baud rate for the serial link between the device and the modem. The baud rate can be 9600, 19200, 38400, 57600, or 115200 bps. The default is 115200 bps.

Example: The following command sets a maximum baud rate of 56Kbps for the serial link:

set modem speed 57600

state

get modem state

state	Shows modem coltrol state, machine state, and HDLC status.

stats

get modem stats

stats	Shows modem status. Displays modem and HDLC layer statistics and the IN table and OUT table statistics.

stop

exec modem stop

stop	Disconnects the current connections and brings down the serial interface.

multicast-group-policy

Use the **multicast-group-policy** commands to define a policy that allows multicast control traffic to cross the security device.

Syntax

get

get multicast-group-policy between *zone1 zone2*

set

```
set multicast-group-policy
    from zone1
      {
      mgroup { mcst_addr1/mask | any }
        to zone2 [ mgroup ]
          [
          igmp-message |
          pim-message
            {
            bsr-static-rp [ join-prune ] |
            join-prune
            }
              [ bi-directional ]
          ] |
      mgroup-list id_num
        to zone2
          {
          igmp-message |
          pim-message
            {
            bsr-static-rp [ join-prune ] |
            join-prune
            }
              [ bi-directional ] ]
          }
```

Keywords and Variables

between

get multicast-group policy between *zone1 zone2*

between	Displays the multicast policy configured between the specified zones.

bi-directional

set multicast-group policy from { ... } to { ... } bi-directional
unset multicast-group policy from { ... } to { ... } bi-directional

bi-directional	Specifies that the policy applies to both directions of multicast traffic.

Example: The following command defines a bi-directional multicast group policy that allows PIM messages between the trust and untrust zones:

set multicast-group-policy from trust mgroup any to untrust pim-message bsr-static-rp join-prune bi-directional

from ... to

set multicast-group policy from *zone1* mgroup *mcst_addr1* to *zone2* mgroup *mcst_addr2* { ... }
set multicast-group policy from *zone1* mgroup any to *zone2* { ... }
set multicast-group policy from *zone1* mgroup-list *id_num* to *zone2*
unset multicast-group policy from *zone1* mgroup *mcst_addr1* to *zone2* { ... }
unset multicast-group policy from *zone1* mgroup any to *zone2*
unset multicast-group policy from *zone1* mgroup-list *id_num* to *zone2*

from { ... } to — Specifies the two zones between which the policy applies.

- *zone1* is the name of the source security zone.
- *zone2* is the name of the destination security zone.
- *mcst_addr1* is the multicast IP address of the multicast group from which the zone accepts multicast packets
- *mcst_addr2* is the translated multicast group address, if you are translating a multicast group address from one zone to another
- *id_num* is the ID number of the access list that specifies the multicast groups from which the zone accepts multicast packets

Example: The following command creates a multicast policy allowing IGMP messages from the Trust zone to the Untrust zone:

set multicast-group-policy from trust mgroup-list 12 to untrust igmp-message

igmp-message

set multicast-group policy from { ... } to { ... } igmp-message
unset multicast-group policy from { ... } to { ... } igmp-message

igmp-message	Specifies a multicast group policy that allows IGMP messages between the specified zones.

pim-message

set multicast-group policy from { ... } to { ... } pim-message
{ bsr-static-rp | join-prune }
unset multicast-group policy from { ... } to { ... } pim-message
{ bsr-static-rp | join-prune }

pim-message	Specifies a multicast group policy that allows PIM BSR and/or join-prune messages between the specified zones.

nrtp

Use the **nrtp** commands to clear all NetScreen Reliable Transfer Protocol (NRTP) packet queues.

NRTP is for multicasting NetScreen Redundancy Protocol (NSRP) control messages to multiple receivers when security devices are in a redundancy cluster (interconnected through the High Availability, or HA, ports). NRTP ensures that the primary security device always forwards configuration and policy messages to the backup devices.

Syntax

clear

```
clear [ cluster ] nrtp queues
```

get

```
get nrtp
    {
    counters ( all | receive [ number ] | send } |
    group |
    xmtq
    }
```

Keywords and Variables

cluster

```
clear cluster nrtp queues
```

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

counters

get nrtp counters (all | receive *number* | send }

counters — Displays statistical information tracked by counters.

- **all** Displays all counter statistics.
- **receive** [*number*] Displays only counter statistics for information that the device receives from other devices in the cluster. The optional *number* parameter is an ID number that identifies a particular device in the cluster.
- **send** Displays only counter statistics for information that the device sends to other devices.

groups

get nrtp group

group	Displays the ID numbers of devices belonging to the group, and a count of the devices in the group.

queues

clear nrtp queues

queues	Clears the NRTP packet queues.

xmtq

get nrtp xmtq

xmtq	Displays the length of the queue containing packets awaiting ACK responses from other devices.

nsgp

Use the **nsgp** commands to configure the GPRS Overbilling Attack notification feature on the Gi firewall (the server).

An Overbilling attack can occur in various ways. It can occur when a legitimate subscriber returns his IP address to the IP pool, at which point an attacker can hijack the IP address, which is vulnerable because the session is still open. When the attacker takes control of the IP address without being detected and reported, the attacker can download data for free (or, more accurately, at the expense of the legitimate subscriber) or send data to other subscribers.

An Overbilling attack can also occur when an IP address becomes available and gets reassigned to another MS. Traffic initiated by the previous MS might be forwarded to the new MS, therefore causing the new MS to be billed for unsolicited traffic.

Syntax

clear

clear nsgp { *ip_addr* | all }

get

get nsgp [detail]

set

set nsgp
{
context *id_num* type session zone *zone* |
md5-authentication *password* |
port *port_num*
}

Keywords and Variables

all

clear nsgp all

all	Closes all active connections on the security device. You can also close active connections on a per IP address basis by entering a specific IP address instead of the keyword **all**.

context

set nsgp context *id_num* type *string* zone *zone*
unset nsgp context *id_num*

context — Creates or deletes a context of a specific type for the specified zone.

- **type** *string* Identifies the type of context. Currently security devices only supports the "session" type.
- **zone** *name* Identifies the zone for which you are creating the context.

The same context must exist on both the client and the server.

detail

get nsgp [detail]

detail	Displays NSGP settings and status of contexts within the current root or virtual system. At the root level, this command also displays information for all virtual systems.

md5-authentication

set nsgp md5-authentication *password*
unset nsgp md5-authentication

md5-authentication — Directs the Gi firewall to enforce the MD5 auth option specified in the TCP header. You can only specify one MD5 authentication password per security device.

This command is only available at the root level and not at the vsys level.

port

set nsgp port *port_num*
unset nsgp port

port — Identifies the port number used by the Gi firewall to receive Overbilling Attack notifications. The default port number is 12521.

This command is only available at the root level and not at the vsys level.

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 the NSM Agent, which resides on the security device. The NSM Agent receives configuration parameters from the management system and pushes it to ScreenOS. The NSM Agent also monitors the device and transmits reports back to the management system.

For more information, refer to the information 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
    {
    bulkcli reboot-timeout { number | disable } |
    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

all

unset nsmgmt all

all	Unsets all NetScreen-Security Manager management configurations.

bulkcli

set nsmgmt bulkcli reboot-timeout { *number* | disable }
unset nsmgmt bulkcli reboot-timeout { *number* | disable }

bulkcli	Enables, disables or sets the bulk-CLI reboot timeout value (expressed in seconds). This setting determines how the device performs rollback when a NSM connection drops during an update session. When this happens, the Agent iterates through all the configured NSM servers once to see if it can establish another connection. If not, the agent waits for the specified time period before it reboots the device to roll back the configuration. The range for the reboot-timeout value is 60 through 86400.

enable

get nsmgmt enable
set nsmgmt enable
unset 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*
unset nsmgmt init {...}

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.

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 on such attacks, see "zone" on page 695.
- **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.
- **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.

proto-dist

get nsmgmt proto-dist { table { bypes | packets } | user-service }
set nsmgmt report proto-dist { ... }
unset nsmgmt report proto-dist { *string* }

proto-dist	Sets or displays parameters for transmission of messages concerning protocol distribution parameters. The categories of protocol distribution are as follows: ■ **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** Displays the user services that are configured on each protocol.

server

set nsmgmt server { primary | secondary }
{ *name_str* | *ip_addr* } [port *number* | src-interface]
unset nsmgmt server { primary | secondary } { *name_str* | *ip_addr* }

server	Identifies the Security Management system server.

nsrp

Use the **nsrp** commands to assign a security device to a failover cluster and to create and configure a virtual security device (VSD) group for the cluster.

The purpose of a VSD group is to allow failover between two or more security devices within a defined cluster. Each VSD group represents a group of devices in a cluster, elects a primary device from the cluster, and provides a virtual security interface (VSI) that external devices use to reference the devices in the cluster.

A group may contain every device in the cluster. For example, if you give three devices the same cluster ID, you can create a VSD group containing all three devices. A device can be in more than one VSD group at a time. For example, a device can be the primary in one VSD group while serving as a backup in another.

To set up a failover VSD group, perform the following steps:

1. Set up a cluster of devices using the **set nsrp cluster** command. This command assigns an identical cluster ID to each device.
2. Set up a VSD group for the cluster using the **set nsrp vsd-group** command.
3. Set up a VSI for the VSD group using the **set interface** command.

Syntax

clear

```
clear [ cluster ] nsrp counter
    [
    packet-fwd |
    protocol |
    rto
    ]
```

exec

```
exec nsrp
    {
    probe interface [ mac_addr ] [ count number ] |
    sync
      {
      file [ name filename ] from peer |
      global-config [ check-sum | save ] |
      rto
        {
        all |
        arp |
        attack-db |
        auth-table |
        dhcp |
```

```
dip-in |
dns |
h323 |
l2tp |
phase1-SA |
pki |
rm |
rpc |
session |
vpn |
}
   { from peer }
} |
vsd-group grp_num mode { backup | ineligible | init | pb }
}
```

get

```
get nsrp
   [
   cluster |
   counter [ packet-fwd | protocol | rto ] |
   group |
   ha-link |
   monitor [ all | interface | track-ip | zone ] |
   rto-mirror |
   track-ip [ ip ip_addr ] |
   vsd-group [ id id_num | all ]
   ]
```

set

```
set nsrp
   {
   arp number |
   auth password pswd_str |
   cluster [ id number | name name_str ] |
   config sync |
   data-forwarding |
   encrypt password pswd_str |
   ha-link probe [ interval number ] [ threshold number ] |
   interface interface |
   link-hold-time number |
   link-up-on-backup |
   monitor
      {
      interface interface [ weight number ] |
      threshold number |
      track-ip
         {
         ip
         [ ip_addr
            [
            interface interface |
            interval number |
            method { arp | ping } |
            threshold number |
```

```
            weight number
            ]
        ] |
        threshold number |
        weight number |
        } |
    zone zone [ weight number ]
    } |
rto-mirror
    {
    hb-interval number |
    hb-threshold number |
    id id_num { direction { in | out } } |
    session [ ageout-ack | non-vsi | off ] |
    sync
    }
secondary-path interface |
track-ip
    [
    ip
        [ ip_addr
            [
            interface interface |
            interval number |
            method { arp | ping } |
            threshold number |
            weight number
            ]
        ]
    threshold number
    ] |
vsd-group
    {
    id id_num
        [
        mode ineligible |
        preempt [ hold-down number ] |
        priority number
        ] |
    hb-interval number |
    hb-threshold number |
    init-hold number |
    master-always-exist
    }
}
```

Keywords and Variables

arp

set nsrp arp *number*
unset nsrp arp *number*

arp	Sets the number of gratuitous Address Resolution Protocol (ARP) requests that a newly elected primary unit sends out, notifying other network devices of its presence. The default is 4.

Example: The following command instructs the security device to send out seven ARP requests:

set nsrp arp 7

auth

set nsrp auth password *pswd_str*
unset nsrp auth

auth	Instructs the security device to authenticate NSRP communications using the specified password. Valid passwords contain from 1 to 15 characters.

Example: The following command sets the NSRP authentication password to *swordfish*:

set nsrp auth password swordfish

cluster

get nsrp cluster
set nsrp cluster id *number*

cluster id	Assigns the security device to a cluster, expressed as an integer (from 1 to 7, inclusive) to identify the cluster.

Example: The following command assigns the security device to cluster 2:

set nsrp cluster id 2

cluster (clear)

clear cluster nsrp counter [...]

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

config sync

set nsrp config sync
unset nsrp config sync

config sync	Enables or disables synchronization of device configuration between members of the NSRP cluster. After you enable this setting, any configuration change automatically propagates to the other devices in the cluster.

counter

clear [cluster] nsrp counter [packet-fwd | protocol | rto]
get nsrp counter [packet-fwd | protocol | rto]

counter — Clears or displays the NSRP counter values.

- **packet-fwd** Clears or displays packet-forwarding counters only.
- **protocol** Clears or displays NSRP protocol counters only.
- **rto** Clears or displays RTO message counters only.

data-forwarding

set nsrp data-forwarding
unset nsrp data-forwarding

data-forwarding	Enables of disables packet forwarding. The default setting is enabled.

encrypt password

set nsrp encrypt password *pswd_str*
unset nsrp encrypt

encrypt password	Specifies that NSRP communications be encrypted using the specified password. Valid passwords contain from 1 to 15 characters.

Example: The following command sets the NSRP encryption password to *manta*:

set nsrp encrypt password manta

group

get nsrp group

group	Displays information on the VSD group.

ha-link probe

set nsrp ha-link probe [interval *number*] [threshold *number*]
unset nsrp ha-link probe [interval] [threshold]

ha-link probe	Specifies the automatic sending of NSRP probe requests on all interfaces that are bound to the HA zone. If a reply is received from the peer within the threshold, the HA link is considered to be up. If the number of consecutive probe requests sent without receiving a reply from the peer reaches or exceeds the threshold, the HA link is considered to be down. You can specify the following optional parameters: ■ **interval** *number* Specifies the interval, in seconds, at which probe requests are sent. Enter a number between 0 and 255. If you do not specify an interval, probe requests are sent every second. ■ **threshold** *number* Specifies the failure threshold for the HA link. If the number of consecutive probe requests sent without receiving a reply from the peer reaches or exceeds the threshold, the HA link is considered to be down. Enter a value between 0 and 255. The default threshold is 5.

interface

set nsrp interface *interface*

interface	The name of the interface to serve as the high-availability port. For information on interfaces, see "Interface Names" on page A-I.

Example: The following command specifies that the NSRP interface is ethernet4:

set nsrp interface ethernet4

link

get nsrp link

link	Displays HA link information

link-hold-time

set nsrp link-hold-time *number*
unset nsrp link-hold-time

link-hold-time	The delay time (in seconds) before the security device brings up the link with the peer device.

link-up-on-backup

set nsrp link-up-on-backup
unset nsrp link-up-on-backup

link-up-on-backup	Specifies that the link is always up on the backup device.

monitor

get nsrp monitor [zone | interface | track-ip] [all]
set nsrp [vsd-group id *id_num*] monitor { ... }
unset nsrp [vsd-group id *id_num*] monitor { ... }

monitor	Specifies monitoring of NSRP objects (a physical interface, a zone, or tracked IP addresses) to determine VSD or device failure. You can specify the following parameters:

- **vsd-group id** *id_num* Identifies the virtual security device (VSD) to which the threshold or monitored objects you configure applies. If you do not specify a VSD, the threshold or monitored objects you configure apply to the entire device.
- **all** Displays monitoring information for the device and all VSDs. If you specify **vsd-group id**, only monitoring information for the VSD is displayed.
- **interface** *interface* [**weight** *number*] Identifies the interface to be monitored and the weight that failure of the interface contributes to the failover threshold. The default weight is 255.
- **threshold** *number* Specifies a failover threshold which determines the failure of a specific VSD (if a VSD is specified) or failure of the device (if no VSD is specified). If the cumulative weight of the failure of all monitored objects (a physical interface, a zone, or tracked IP addresses) exceeds the threshold, the VSD or the device fails. The default threshold value is 255.
- **track-ip weight** *number* [**threshold** *number*] [**ip** *ip_addr*] Enables tracked IP object monitoring and the weight that failure of the tracked IP object (all tracked IP addresses) contributes to the device or VSD failover threshold. The default **weight** value is 255. The **threshold** value is the total weight of failed tracked IP addresses that determines failure of the tracked IP object. The default **threshold** value is 255. Specifies monitoring of tracked IP addresses to determine VSD or device failure. For each ip *ip_addr*, you can configure the following:
  - **interface** *interface* Specifies the outgoing interface through which the security device performs tracking. for the specified IP address. If you do not specify an interface, ping tries to find an outgoing interface from routing table entries and ARP tries to find an outgoing interface within the same subnet. If an interface is not found, the tracking attempt fails.
  - **interval** *number* Specifies the interval, in seconds, between ping or ARP attempts to the specified IP address. Enter a value between 1 and 200. The default is 1.
  - **method { arp** | **ping }** Specifies the method used for tracking the specified IP address. The default is ping.
  - **threshold** *number* Defines the number of failed tracking attempts that can occur before the tracking of the specified IP address is considered failed. The default is 3.
  - **weight** *number* Defines the weight of the failed tracking of the specified IP address. The default is 1.
- **zone** *zone* [**weight** *number*] Identifies the zone to be monitored and the weight that failure of all physical interfaces in the zone contributes to the failover threshold. The default weight is 255.

probe

exec nsrp probe *interface* [*mac_addr*] [count *number*]

probe	Directs the device to immediately begin sending an NSRP probe request every second on an HA zone interface, for the number of times specified by **count**. If the peer receives a reply, the HA link is considered to be up. (If the request times out before the peer receives a reply, the HA link is considered to be down.) The device takes no action if there is no reply. (See ha-link probe on page 386.) ■ *interface* Identifies the HA zone interface on which probe requests are sent. You must specify an interface that is bound to the HA zone. ■ *mac_addr* Identifies the destination MAC address of an HA interface on a peer device. If you do not specify a destination MAC address, the device uses the default NSRP MAC address to send the probe request. ■ **count** *number* Specifies the number of times that the device sends the probe request. Enter a number greater than or equal to 1. (The default is 1.)

rto-mirror

get nsrp rto-mirror
set nsrp rto-mirror { ... }
unset nsrp rto-mirror { ... }

rto-mirror	Creates an optional RTO mirror between two devices in a VSD group to back up runtime objects (RTOs). In most cases, using this option is not necessary. Normally, RTO objects synchronize after execution of the **set nsrp rto sync** command. A security device can belong to only one RTO mirror group at a time. ■ **id** *id_num* Identifies the VSD group using its identification number *id_num*, an integer value between 1 and 127 inclusive. The **direction** setting determines whether the RTO mirror group direction is inbound or outbound. ■ **hb-interval** *number* Specifies the heartbeat interval in seconds. ■ **hb-threshold** *number* Specifies the heartbeat-lost threshold. The minimum threshold value is 16 heartbeats. ■ **session ageout-ack** Specifies a time value based on which the backup device sends an ack message to the primary device to refresh its sessions or time them out. The session age-out value of a backup device is eight times that of the primary device. ■ **session non-vsi** Enables the synchronization of non-VSI sessions. ■ **session off** Disables the RTO session. ■ **sync** Enables RTO object synchronization.

Example: The following command specifies that the RTO mirror group (10) direction is inbound:

set nsrp rto-mirror id 10 direction in

secondary-path

set nsrp secondary-path *interface*
unset nsrp secondary-path

secondary-path	Specifies a secondary NSRP link interface.

Example: The following command specifies that the secondary NSRP link interface is *ethernet5*:

set nsrp secondary-path ethernet5

sync

exec nsrp sync { ... }

sync

Specifies the name of a particular configuration, file, or RTO to copy from one unit to the other.

- **file** Specifies synchronization of the files in flash memory.
  - **name** *filename* specifies a particular file in flash memory. (Executing the **file** option without specifying a filename copies all the files.)
  - **from peer** specifies all files from the peer device.
- **global-config** Specifies synchronization of the current device configurations. The check-sum switch compares the checksum after synchronization. The save switch synchronizes the PKI configuration and saves the synchronization configuration to flash memory.
- **rto** Specifies synchronization of the current runtime objects (RTOs) in the RTO mirror.
  - **all** Specifies all possible realtime objects.
  - **arp** Specifies the Address Resolution Protocol (ARP) information.
  - **attack-db** Specifies the Deep Inspection (DI) attack database table information.
  - **auth-table** Specifies the authentication table information.
  - **dhcp** Specifies Dynamic Host Configuration Protocol (DHCP) information.
  - **dip-in** Specifies information on the incoming dynamic Internet Protocol (DIP) addresses table.
  - **dns** Specifies the Domain Name System (DNS) information.
  - **h323** Specifies H.323 information.
  - **pki** Specifies certificate information.
  - **phase1-sa** Specifies information on IKE Phase-1 Security Associations (SAs)
  - **rm** Specifies Resource Manager information.
  - **rpc** Specifies information on Remote Procedure Call (RPC) mapping.
  - **session** Specifies the session information.
  - **vpn** Specifies all virtual private network (VPN) information.

Example: The following command instructs the security device to synchronize all runtime objects:

exec nsrp sync rto all from peer

track-ip

get nsrp track-ip [ip *ip_addr*]
set nsrp track-ip [...]
unset nsrp track-ip [...]

track-ip

Enables path tracking, which is a means for checking the network connection between an interface and that of another device. The IP address *ip_addr* identifies the other network device to check.

Executing **unset nsrp track ip** resets the track options to their default values.

- **ip** *ip_addr*
  - **interface** *interface* Specifies the interface through which the security device performs the path tracking. If you do not specify an interface, the device automatically chooses the interface for IP tracking using either the ping or ARP method. If ping is used, the device tries to find an outgoing interface from entries in the routing table. If ARP is used, the device tries to find an outgoing interface within the same subnet. If an interface is not found, the tracking attempt fails.
  - **interval** *number* Specifies the interval in seconds between path tracking attempts. Required value is between 1 and 200. The default is 1.
  - **method { arp | ping }** Specifies the method used for path tracking. The default is *ping*.
  - **threshold** *number* Defines the number of failed tracking attempts that can occur before the tracking of the IP address is considered failed. The default is 3.
  - **weight** *number* Defines the path weight. Valid weights are between 1 and 255 inclusive. The default weight is 1.
- **threshold** *number* Defines the number of failed tracking attempts that can occur before the device fails over. The default is 255.
- **weight** *number* Defines the sum of the weights of the tracked IP addresses that determine failover. The default is 255.

Example: The following command enables path tracking through interface ethernet4 to a device at IP address 172.16.10.10:

set nsrp track-ip ip 172.16.10.10 interface ethernet4

vsd-group

get nsrp vsd-group [id *id_num* | all]
set nsrp vsd-group [...]
unset nsrp vsd-group [...]

vsd-group

Configures a VSD group for a cluster.

id *id_num*

Creates a VSD group, identified by *id_num* (from 1 to 8, inclusive), that contains all members belonging to a single cluster of devices. Once created, a VSD group elects a primary unit from the cluster it contains. Other devices reference the device cluster in the VSD group through the group's virtual security identification (VSI).

- **mode ineligible** Determines the running mode of the security device. The **ineligible** switch specifies that the local device is not intended for failover, even after system restart. (This may be necessary for administrative reasons.) Executing **unset nsrp vsd-group id** *number* **mode ineligible** specifies that the device is eligible again.
- **preempt** [**hold-down** *number*] Determines if the primary unit keeps its primary status until the unit itself relinquishes that status. To prevent rapid failovers, the primary device waits for the specified hold-down interval, expressed as a number between 0 to 600 seconds inclusive. The default is 3.
- **priority** *number* The priority level of the device, expressed as an integer from 1 to 254, inclusive. The priority level determines the failover order for the device. The failover order determines which unit is the primary unit when two security devices in a redundant group power up simultaneously, and which backup unit becomes the next primary during a failover. (The unit with the number closest to 1 becomes the primary unit.)
- **init-hold** The number of heartbeats that occurs before the system exits the initial state (init mode). This value can be an integer from 5 to 255. The default is 5.
- **hb-interval** *number* Specifies the heartbeat interval, expressed in milliseconds. This value can be an integer from 200 to 1000. The default is 1000.
- **hb-threshold** *number* Specifies the heartbeat-lost threshold, the number of lost heartbeats allowed before failure. This value can be an integer from 3 to 255. The default is 3.
- **master-always-exist** Directs the system to elect a primary unit and keep it operative even if all units in the NSRP cluster fail (by monitoring result). For example, if you disable master-always-exist, and two units tracking an IP later fail due to monitoring results, both units become inoperable and traffic cannot go through. If you enable **master-always-exist**, and both units fail, the cluster still elects a primary unit, which remains operable, thus allowing traffic through.

Example 1: The following command disables the local device for failover:

set nsrp vsd-group id 2 mode ineligible

Example 2: The following command specifies that ten heartbeats must occur before the device exits the initial state:

set nsrp vsd-group init-hold 10

vsd-group (exec)

exec nsrp vsd-group *grp_num* mode { ... }

vsd-group *grp_num* mode	Specifies a VSD group and the security device's new mode. ■ In **backup** mode, the device works for the primary device when the primary device fails. ■ In **ineligible** mode, the device is unavailable as a backup for the primary device. ■ In **init** mode, the device is in the transient state that occurs when it joins the VSD group. (At the end of this initial hold up time, the device transitions to another state, such as primary, backup, or primary backup.) ■ In **pb** (primary backup) mode, the unit is the first to take over when the primary unit fails.

Example: The following command instructs the security device to take over when the primary unit fails:

exec nsrp vsd-group 2 mode pb

Defaults

The default value of **preempt** [**hold-down** *number*] is *zero*.

The default value of **vsd-group id** *id_num* **priority** *number* is *100*.

The default value of **vsd-group id** *id_num* **hb-interval** *number* is *1000* (1,000 milliseconds, or one second).

Creating an NSRP Cluster

The following commands set up an NSRP cluster consisting of two security devices

- Two VSD groups for the cluster
- VSI for the VSD group
- RTO object synchronization enabled, including session synchronization

On Device A

Trust Zone Redundant Interface and Manage IP

```
set interface redundant2 zone trust
set interface ethernet2/1 group redundant2
set interface ethernet2/2 group redundant2
set interface redundant2 manage-ip 10.1.1.3
```

Cluster and VSD Groups

```
set nsrp cluster id 1
set nsrp vsd-group id 0 preempt hold-down 10
set nsrp vsd-group id 0 preempt
set nsrp vsd-group id 0 priority 1
set nsrp vsd-group id 1
set nsrp monitor interface redundant2
set nsrp rto-mirror sync
save
```

On Device B

Trust Zone Redundant Interface and Manage IP

set interface redundant2 zone trust
set interface ethernet2/1 group redundant2
set interface ethernet2/2 group redundant2
set interface redundant2 manage-ip 10.1.1.4

Cluster and VSD Groups

set nsrp cluster id 1
set nsrp rto-mirror sync
set nsrp vsd-group id 1 priority 1
set nsrp vsd-group id 1 preempt hold-down 10
set nsrp vsd-group id 1 preempt
set nsrp monitor interface redundant2
set nsrp arp 4
set arp always-on-dest

Untrust Zone Redundant Interface

set interface redundant1 zone untrust
set interface ethernet1/1 group redundant1
set interface ethernet1/2 group redundant1

Virtual Security Interfaces

set interface redundant1 ip 210.1.1.1/24
set interface redundant2 ip 10.1.1.1/24
set interface redundant1:1 ip 210.1.1.2/24
set interface redundant2:1 ip 10.1.1.2/24

Routes

set vrouter untrust-vr route 0.0.0.0/0 interface redundant1 gateway 210.1.1.250
set vrouter untrust-vr route 0.0.0.0/0 interface redundant1:1 gateway 210.1.1.250
save

ntp

Use the **ntp** commands to configure the security device for Simple Network Time Protocol (SNTP).

As its name implies, SNTP is a simplified version of Network Time Protocol (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
      {
      ip_addr | dom_name |
      backup1
        {
        ip_addr | dom_name |
        src-interface interface |
        key-id number preshare-key string
        } |
      backup2
        {
        ip_addr | dom_name |
        src-interface interface |
        key-id number preshare-key string
        } |
      key-id number preshare-key string |
      src-interface interface
      }
    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** 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** 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—the same as for 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 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*
set ntp server { backup1 | backup2 } src-interface *interface*
unset ntp server { ... }

server

- *ip_addr* The IP 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 IP 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 backup server for authentication purposes.
  - **preshare key** Assigns a preshared key to the backup server for authentication purposes.
  - **src-interface** *interface* Indicates the source interface the device uses to send NTP requests to the backup server.
- **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.
- **src-interface** *interface* Indicates the source interface the device uses to send NTP requests.

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 Greenwich Mean Time (GMT). *number2* expresses minutes.

Example: The following command sets the time zone to GMT:

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

get os { cost | flow | kernel | misc | task [*name_str* | *id_num*] }

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* | *id_num*]

task	Displays information on a specified task (*name_str*) or task id (*id_num*).

OSPF Commands

Use the **ospf** context to begin configuring Open Shortest Path First (OSPF) routing protocol for a virtual router.

Context Initiation

Initiating the **ospf** 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. Set the router ID for this virtual routing instance.

   set router-id { *id_num* | *ip_addr* }

   For example:

   device(trust-vr)-> **set router-id 172.16.10.10**

3. Enter the **ospf** context by executing the **set protocol ospf** command.

   device(trust-vr)-> **set protocol ospf**

4. Enable OSPF protocol (it is *disabled* by default).

   device(trust-vr/ospf)-> **set enable**

OSPF Command List

The following commands are executable in the **ospf** context. Click on a keyword in the table to go to complete syntax and usage information.

advertise-def-route	Use the **advertise-def-route** commands to advertise or display the default route of the current virtual routing instance (0.0.0.0/0) in all areas. Every router has 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**
area	Use the **area** commands to configure an area for an OSPF virtual routing instance. An OSPF area is a region that contains a collection of routers or virtual routing instances. Command options: **get, set, unset**

authentication	Use the **authentication** command to display authentication for the OSPF virtual routing instance. Command options: **get**
auto-vlink	Use the **auto-vlink** commands to direct the local virtual router to automatically create virtual links. Using automatic virtual links replaces the more time-consuming process of creating each virtual link manually. A virtual link is a conveyance that enables two unconnected segments that cannot reach a backbone router to connect with each other. Command options: **get, set, unset**
config	Use the **config** command to display all commands executed to configure the OSPF local virtual routing instance. Command options: **get**
database	Use the **database** command to display details about the current OSPF link state database. Command options: **get**
enable	Use the **enable** commands to enable or disable OSPF from the current routing instance. Command options: **set, unset**
hello-threshold	Use the **hello-threshold** commands to set or display the hello threshold. When a neighbor device exceeds this threshold by flooding the virtual router with hello packets, the virtual router drops the extra packets. A Hello packet is a broadcast message that announces the presence of a routing instance on the network. Command options: **get, set, unset**
interface	Use the **interface** command to display all OSPF interfaces on the virtual router. Command options: **get**
lsa-threshold	Use the **lsa-threshold** commands to set or display the Link State Advertisement (LSA) threshold. When a neighbor device exceeds this threshold by flooding the virtual router with LSA packets, the virtual router drops the extra packets. Link State Advertisements (LSAs) enable OSPF routers to make device, network, and routing information available for the link state database. Command options: **get, set, unset**
neighbor	Use the **neighbor** command to display details about neighbor devices. Command options: **get**
redistribute	Use the **redistribute** commands to import routes from a different protocol than the one used by the current virtual routing instance. The types of routing protocols from which to import routes include: ■ manually-created routes (static) ■ routes from BGP (bgp) ■ routes that have at least one interface with an IP address assigned to it (connected) ■ routes from RIP (rip) ■ routes that have already been imported (imported). Command options: **set, unset**

reject-default-route Use the **reject-default-route** commands to reject or restore the default route learned from OSPF (0.0.0.0/0) in the current routing instance.

Every router has a default route entry in its routing table. This default route matches every destination. (Any entry with a more specific prefix overrides the default route entry.)

Command options: **get, set, unset**

retransmit Use the **retransmit** commands to retransmit packets before adjacency ends.

Command options: **set, unset**

rfc-1583 Use the **rfc-1583** commands to use routing table calculation methods consistent with standards specified in the Request For Comments 1583 document.

Command options: **get, set, unset**

routes-redistribute Use the **routes-redistribute** command to display details about routes imported from a protocol other than OSPF.

Command options: **get**

rules-redistribute Use the **rules-redistribute** command to display conditions set for routes imported from a protocol other than OSPF.

Command options: **get**

statistics Use the **statistics** command to display information about Hello packets, link state packets, database descriptions, Shortest Path First (SPF) packets, packets dropped, errors, and other traffic statistics related to the current OSPF virtual routing instance.

Command options: **get**

stub Use the **stub** command to display details about a stub area created in the current OSPF virtual routing instance.

Command options: **get**

summary-import Use the **summary-import** commands to summarize a route redistribution.

After importing a series of routes to the current OSPF routing instance from a router running a different protocol, you can bundle the routes into one generalized (or summarized) address that uses the same network stem of the prefix address. By summarizing multiple addresses, you allow the OSPF routing instance to treat a series of routes as one route, thus simplifying the process.

Command options: **get, set, unset**

vlink Use the **vlink** commands to create a virtual link for the current routing instance.

A virtual link is a conveyance that allows two segments to connect when the backbone router bridging them cannot reach either segment.

Command options: **get, set, unset**

vneighbor Use the **vneighbor** command to display information about a virtual routing instance neighbor.

Command options: **get**

advertise-def-route

Use the **advertise-def-route** commands to advertise or display the default route of the current virtual routing instance (0.0.0.0/0) in all areas.

Every router has a default route entry, which matches every destination. Any route entry with a more specific prefix than the default route entry overrides the default entry.

Before you can execute the **advertise-def-route** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get advertise-def-route

set

```
set advertise-def-route
    {
    always metric number [ preserve-metric ] |
    metric number | preserve-metric
    }
      metric-type { 1 | 2 }
```

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**, only a non-OSPF active default route is advertised. 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-type { 1 | 2 }

metric	Specifies the metric (cost), which indicates the overhead associated with the default route. Enter a number between 1-15. You must specify this parameter if you specify the **always** option.

metric-type

set advertise-def-route [always] metric *number* metric-type { 1 | 2 }

metric-type	Specifies the external route type to determine path preference. ■ **1** Directs the routing instance to use a Type 1 route to evaluate the default route. A type 1 route is a comparable route, with a lower cost than a type 2 route. ■ **2** Directs the routing instance to use a Type 2 route to evaluate the default route. A type 2 route is a noncomparable route, with a higher cost than a type 1 route.

preserve-metric

set advertise-def-route [always] preserve-metric metric-type { 1 | 2 }

preserve-metric	Instructs the security device to use the original (source) route metric when the route is redistributed.

area

Use the **area** commands to configure an area for an OSPF virtual routing instance.

An OSPF area is a region that contains a collection of routers or virtual routing instances.

Before you can execute the **area** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get area [*id_num* | *ip_addr*]

set

set area { *id_num* | *ip_addr* }
  [
  metric-default-route *number* |
  no-summary |
  nssa |
  range *ip_addr/mask* { advertise | no-advertise } |
  stub |
  type-default-route { 1 | 2 }
  ]

Keywords and Variables

Variable Parameters

get area [*id_num* | *ip_addr*]
set area { *id_num* | *ip_addr* }
unset area { *id_num* | *ip_addr* }

ip_addr	The IP address that identifies the area.
id_num	The OSPF area ID that identifies the area.

metric-default-route

set area *id_num* metric-default-route *number*
unset area *id_num* metric-default-route *number*

metric-default-route	(NSSA and stub areas only) Specifies the metric for the advertised default route. The default metric is 1. Enter a number between 1-65535.

no-summary

set area *id_num* no-summary
unset area *id_num* no-summary

no-summary	(NSSA and stub areas only) Prevents summary LSAs from being advertised into the area. By default, summary LSAs are advertised into the area.

nssa

set area *id_num* nssa
unset area *id_num* nssa

nssa	Specifies that the area is a “not so stubby area.”

range

set area *id_num* range *ip_addr/mask* { advertise | no-advertise }
unset area *id_num* range *ip_addr/mask*

range	(All areas) Summarizes a specified range of IP addresses in summary LSAs. You can specify multiple ranges for the area. You can specify whether the summarized addresses are advertised inside the area or not with the **advertise** and **no-advertise** keywords.

stub

set area *id_num* stub
unset area *id_num* stub

stub	Specifies the area is a stub area.

type-default-route

set area *id_num* type-default-route { 1 | 2 }
unset area *id_num* type-default-route { 1 | 2 }

type-default-route	(NSSA area only) Specifies the external metric type for the default route. The default metric type is 1. Specify either **1** or **2**.

authentication

Use the **authentication** command to display authentication information for the OSPF virtual routing instance.

Before you can execute the **authentication** command, you must initiate the **ospf** context. (See Context Initiation on page 401.)

Syntax

get authentication

Keywords and Variables

None.

auto-vlink

Use the **auto-vlink** commands to automatically create or display details about virtual links.

Using automatic virtual links replaces the more time-consuming process of creating each virtual link manually. A virtual link is a conveyance that enables two unconnected segments that cannot reach a backbone router to connect with each other.

Before you can execute the **auto-vlink** commands, you must initiate the **ospf** context. (See Context Initiation on page 401.)

Syntax

get

get auto-vlink

set

set auto-vlink

Keywords and Variables

None.

config

Use the **config** command to display all commands executed to configure the OSPF local virtual routing instance.

Before you can execute the **config** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

get config

Keywords and Variables

None.

database

Use the **database** command to display details about the current OSPF database.

Before you can execute the **database** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

get database
    [detail] [area [number | *ip_addr*]]
        [asbr-summary | external | network | nssa-external | router | summary
            [
            adv-router *ip_addr* |
            self-originate
            ]
                [link-state-id *ip_addr*]
        ]

Keywords and Variables

adv-router

get database [...] adv-router *ip_addr* [...]

adv-router	Displays the LSAs (Link State Advertisements) from the specified advertising router (*ip_addr*).

Example: The following command displays the LSAs from a router with router ID 172.16.10.10:

get database adv-router 172.16.10.10

area

get database [...] area [*number* | *ip_addr*] [...]

area	Displays the LSAs in the current area.

Example: The following command displays the LSAs from an area (4):

get database area 4

detail

get database detail [...]

detail	Displays detailed information.

Example: The following command generates a detailed display of LSAs from an area (4):

get database detail area 4

external

get database [...] external [...]

external	Displays external LSAs.

Example: The following command displays external LSAs:

get database external

link-state-id

get database { ... } link-state-id *ip_addr*

link-state-id	Displays the LSA with a specified link-state ID (*ip_addr*).

Example: The following command generates a detailed display of external LSAs with link-state ID *172.16.1.1*:

get database detail external link-state-id 172.16.1.1

network

get database [...] network [...]

network	Displays the network LSAs.

Example: The following command displays network LSAs:

get database network

nssa-external

get database [...] nssa-external [...]

nssa-external	Displays the not-so-stubby areas (NSSAs) external LSAs.

Example: The following command displays external LSAs for not-so-stubby areas:

get database nssa-external

router

get database [...] router [...]

router	Displays router LSAs.

Example: The following command displays router LSAs:

get database router

self-originate

get database [...] self-originate [...]

self-originate	Displays self-originated LSAs.

Example: The following command displays self-originated LSAs:

get database self-originate

summary

get database [...] summary [...]

summary	Displays summary LSAs.

Example: The following command displays summary LSAs:

get database summary

enable

Use the enable commands to enable or disable OSPF from the current routing instance.

Before you can execute the **set enable** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

set enable

Keywords and Variables

None.

hello-threshold

Use the **hello-threshold** commands to set or display the hello threshold. When a neighbor device exceeds this threshold by flooding the virtual router with hello packets, the virtual router drops the extra packets. A hello packet is a broadcast message that announces the presence of a routing instance on the network.

Before you can execute the **hello-threshold** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get hello-threshold

set

set hello-threshold *number*

Keywords and Variables

Variable Parameter

set hello-threshold *number*

number	The maximum number of hello packets the virtual router accepts from a neighbor in the hello interval.

Example: The following command sets the maximum number of packets to allow in the hello interval to 1000:

device(trust-vr/ospf)-> **set hello-threshold 1000**

interface

Use the **interface** command to display all OSPF interfaces on the virtual router.

Before you can execute the **interface** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get interface

Keywords and Variables

None.

lsa-threshold

Use the **lsa-threshold** commands to set or display the Link State Advertisement (LSA) threshold. When a neighbor device exceeds this threshold by flooding the virtual router with LSA packets, the virtual router drops the extra packets.

Link State Advertisements (LSAs) enable OSPF routers to make device, network, and routing information available for the link state database.

Before you can execute the **lsa-threshold** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get lsa-threshold

set

set lsa-threshold *number1 number2*

Keywords and Variables

Variable Parameters

set lsa-threshold *number1 number2*

number1	The LSA time interval (in seconds).
number2	The maximum number of LSAs that the virtual router accepts within the time interval expressed by *number1*.

Example: The following command creates an OSPF LSA threshold:

set lsa-threshold 10 30

neighbor

Use the **neighbor** command to display details about neighbor devices.

Before you can execute the **neighbor** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get neighbor

Keywords and Variables

None.

redistribute

Use the **redistribute** commands to import known routes from a router running a different protocol than the current virtual routing instance.

The types of routers from which to import routes include:

- Routers with manually created routes (**static**)
- Routers running BGP (**bgp**)
- Routers that have at least one interface with an IP address assigned to it (**connected**)
- Routers with routes that have already been imported (**imported**)
- Routers running RIP (**rip**)

Before you can execute the **redistribute** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get routes-redistribute [summary]
get rules-redistribute

set

set redistribute route-map *string* protocol
{ bgp | connected | discovered | imported | rip | static }

Keywords and Variables

protocol

set redistribute route-map *string* protocol { ... }
unset redistribute route-map *name_str* protocol { ... }

protocol	Specifies routing protocol. The route map can use the protocol type to determine whether to forward or deny an incoming packet. ■ **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 a router that has at least one interface with an IP address assigned to it. ■ **discovered** specifies that the route map performs an daction only on routes discovered by the device. ■ **imported** specifies that the route map performs an action only on imported routes in the subnetwork. ■ **rip** specifies that the route map performs an action only on RIP routes in the subnetwork. ■ **static** specifies that the route map performs an action only on static routes in the subnetwork.

Example: The following command redistributes a route that originated on a router that has at least one interface with an IP address assigned to it:

device(trust-vr/ospf)-> **set redistribute route-map map1 protocol connected**

route-map

set redistribute route-map *string* protocol { ... }
unset redistribute route-map *string* protocol { ... }

route-map	Identifies the route map that indicates the path for which the route should be imported.

Example: The following command redistributes a route that originated from a BGP routing domain into the current OSPF routing domain:

device(trust-vr/ospf)-> **set redistribute route-map map1 protocol bgp**

reject-default-route

Use the **reject-default-route** commands to reject or restore the default route learned from OSPF (0.0.0.0/0).

Every router has a default route entry in its routing table. This default route matches every destination. (Any entry with a more specific prefix overrides the default route entry.)

Before you can execute the **reject-default-route** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get reject-default-route

set

set reject-default-route

Keywords and Variables

None.

retransmit

Use the **retransmit** command to set the number of packets to resend before adjacency ends.

Before you can execute the **retransmit** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

set retransmit { dc *number* | non-dc *number* }

Keywords and Variables

Variable Parameters

set retransmit dc *number*

number	Sets the number of packets to resend before adjacency ends. The retransmit range is between 2 and 240 packets.

Example: The following command shows setting a demand circuit to resend 10 packets prior to the end of the adjacency:

device(trust-vr/ospf)-> **set retransmit dc 10**

dc

set retransmit dc *number*
unset retransmit dc

dc	Indicates that the type of connection is a demand circuit.

non-dc

set retransmit non-dc *number*
unset retransmit non-dc

non-dc	Indicates that the type of connection is not a demand circuit.

rfc-1583

Use the **rfc-1583** commands to use routing table calculation methods consistent with standards specified in RFC 1583.

Before you can execute the **rfc-1583** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get rfc-1583

set

set rfc-1583

Keywords and Variables

None.

routes-redistribute

Use the **routes-redistribute** command to display details about routes imported from a protocol other than OSPF.

Before you can execute the **routes-redistribute** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get routes-redistribute [summary]

Keywords and Variables

summary

get routes-redistribute [summary]

summary	Shows the number of redistributed routes.

rules-redistribute

Use the **rules-redistribute** command to display conditions set for routes imported from a protocol other than OSPF.

Before you can execute the **rules-redistribute** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get rules-redistribute

Keywords and Variables

None.

statistics

Use the **statistics** command to display information about the following objects associated with an OSPF virtual routing instance:

- Hello Packets
- Link State Requests
- Link State Acknowledgments
- Link State Updates
- Database Descriptions
- Areas Created
- Shorted Path First Runs
- Packets Dropped
- Errors Received
- Bad Link State Requests

Before you can execute the **statistics** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

get statistics

Keywords and Variables

None.

stub

Use the **stub** command to display details about a stub area created for the current OSPF virtual routing instance.

Before you can execute the **stub** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

get stub [*ip_addr*]

Keywords and Variables

Variable Parameters

get stub *ip_addr*

ip_addr	Identifies the stub area.

Example: The following command displays details about a stub area created on the current OSPF virtual routing instance:

device(trust-vr/ospf)-> **get stub 192.168.20.20**

summary-import

Use the **summary-import** commands to summarize a route redistribution.

After importing a series of routes to the current OSPF routing instance from a router running a different protocol, you can bundle the routes into one generalized (or *summarized*) address that uses the same network stem of the prefix address. By summarizing multiple addresses, you allow the OSPF routing instance to treat a series of routes as one route, thus simplifying the process.

Before you can execute the **summary-import** commands, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get

get summary-import

set

set summary-import ip *ip_addr/mask* [tag { *ip_addr* | *id_num* }]

Keywords and Variables

ip

set summary-import ip *ip_addr/mask* [...]
unset summary-import ip *ip_addr/mask*

ip	The summarized prefix, consisting of an address (*ip_addr*) and network mask (*mask*) encompassing all the imported routes.

tag

set summary-import ip *ip_addr/mask* tag { *ip_addr* | *id_num* }

tag	A value that acts as an identifier for the summarized prefix. The virtual router uses this identifier when advertising a new external LSA.

Example: The following command summarizes a set of imported routes under one route (20):

device(trust-vr/ospf)-> **set summary-import ip 2.1.1.0/24 tag 20**

vlink

Use the **vlink** commands to create a virtual link for the current routing instance.

A virtual link is a conveyance that allows two segments to connect when the backbone router bridging them cannot reach either segment.

Before you can execute the **vlink** command, you must initiate the **ospf** context. (See “Context Initiation” on page 401.)

Syntax

get

get vlink

set

set vlink area-id { *id_num1* | *ip_addr* } router-id { *id_num2* | *ip_addr* }
  [
  authentication
    {
    active-md5-key-id *id_num* |
    md5 key_str [key-id *id_num*] |
    password *pswd_str*
    } |
  dead-interval *number* |
  hello-interval *number* |
  retransmit-interval *number* |
  transit-delay *number*
  ]

Keywords and Variables

area-id

set vlink area-id *id_num1* { ... }
unset vlink area-id *id_num1* { ... }

area-id	Specifies the ID or IP address of the area through which the virtual link is connected.

authentication

set vlink { ... } authentication { active-md5-key-id | md5 key_str [key-id *id_num*] |
  password *pswd_str* }
unset vlink { ... } authentication [active-md5-key-id | md5 [key-id *id_num*]

authentication	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.

dead-interval

set vlink { ... } dead-interval *number*
unset vlink { ... } dead-interval *number*

dead-interval	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.

hello-interval

set vlink { ... } hello-interval *number*
unset vlink { ... } hello-interval *number*

hello-interval	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.

retransmit-interval

set vlink { ... } retransmit-interval *number*
unset vlink { ... } retransmit-interval *number*

retransmit-interval	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.

router-id

set vlink area-id *id_num1* router-id *id_num2*
unset vlink area-id *id_num1* router-id *id_num2*

router-id	Specifies the ID or IP address of the router at the other end of the virtual link.

Example: The following command creates a virtual link using an area of 0.0.0.1 for router with an ID of 10.10.10.20:

device(trust-vr/ospf)-> **set vlink area-id 0.0.0.1 router-id 10.10.10.20**

transit-delay

set vlink { ... } transit-delay *number*
unset vlink { ... } transit-delay *number*

transit-delay	Specifies the amount of time (in seconds) that elapses before the security device advertises a packet received on the interface.

vneighbor

Use the **vneighbor** command to display information about a neighbor on the virtual link.

Before you can execute the **vneighbor** command, you must initiate the **ospf** context. (See "Context Initiation" on page 401.)

Syntax

get vneighbor

Keywords and Parameters

None.

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)->

password-policy

Use the **password-policy** command to enforce a minimum length and complexity requirement for administrator and authenticated user passwords.

Syntax

get

get password-policy

set

set password-policy user-type
{
admin { complexity-scheme *scheme_id* | minimum-length *number* } |
auth { complexity-scheme *scheme_id* | minimum-length *number* }
}

Arguments

complexity-scheme

set password-policy user-type admin complexity-scheme *scheme_id*

complexity-scheme	Specify one of the following: ■ 0 (zero)—No complexity scheme required. Passwords can contain any combination of alphanumeric characters and are constrained only by minimum-length, if set. ■ 1—Passwords must contain at least two of the following: ■ Uppercase letters ■ Lowercase letters ■ Numbers ■ Nonalphanumeric characters (!@#$%^&*()) A password using the complexity scheme, for example, might be the following: ABcd12&%.

minimum-length

set password-policy user type auth minimum-length *number*
unset password-policy user type auth minimum-length

minimum-length	Specify a minimum length for passwords. The range is 1 to 32, the default is 1.

password-policy

get password-policy
set password-policy { ... }
unset password-policy { ... }

password-policy	A password policy provides centralized password policy enforcement in network environments where a mechanism such as RADIUS authentication is not available or not practical. To view the current password-policy for admin or auth users, enter the **get password-policy** command. To return the security device to the default password settings, use the keyword **unset**.

user-type

set password-policy user-type admin { ... }
unset password-policy user-type admin { ... }
set password-policy user-type auth { ... }
unset password-policy user-type auth { ... }

admin | auth	Specifies whether the password policy applies to a system administrator, or authenticated user.

pbr

Use the **pbr** commands to configure the security device for Policy-Based Routing (PBR). **get** commands allow you to view PBR settings, **set** commands allow you to configure PBR, and **unset** commands allow you to delete or undo a PBR configuration.

See the following keywords for other PBR-related syntax and keywords:

- "access-list" on page 1
- "action-group" on page 3
- "match-group" on page 355
- "policy" on page 455

Syntax

get

get pbr
{
access-list [*ext_acl_id* | configuration] |
action-group [name *action_group_name* | configuration] |
configuration |
match-group [name *match_group_name* | configuration] |
policy [name *policy_name* | configuration]
}

set

set pbr policy
{
name *pbr_policy_name* |
policy *pbr_policy_name* [match *match_group_name*] action *action_group_name* *entry_id*
}

Keywords and Variables

access-list

get pbr access-list [*ext_acl_id* | configuration]

access-list	Shows access-list information. Two keywords allow you to limit or retireve more information: ■ *ext_acl_id* shows information limited to the specified extended access-list. ■ **configuration** shows the complete extended access-list configuration in the virtual router. To configure an extended access-list, refer to the *Concepts & Examples ScreenOS Reference Guide*.

action-group

get pbr action-group [name *action_group_name* | configuration]

action-group	Shows action group information. Two keywords allow you to limit or retireve more information: ■ **name** *action_group_name* shows information limited to the named action group. ■ **configuration** shows the complete action group configuration in the virtual router. To configure an action group, refer to the *Concepts & Examples ScreenOS Reference Guide*.

configuration

get pbr configuration

configuration	Shows the complete PBR configuration within a virtual router.

match-group

get pbr match-group [name *match_group_name* | configuration]

match-group	Shows match group information. Two keywords allow you to limit or retrieve more information: ■ **name** *match_group_name* shows information limited to the specified match group. ■ **configuration** shows the complete match group configuration in the virtual router. To configure a match group, refer to the *Concepts & Examples ScreenOS Reference Guide*.

policy

get pbr policy [name *policy_name* | configuration]
set pbr policy name *pbr_policy_name*
set pbr policy *pbr_policy_name* [match *match_group_name*] action *action_group_name* *entry_id*

policy	Shows access-list information. Two keywords allow you to limit or retrieve more information: ■ **name policy_name** shows information limited to the specified policy-based routing (PBR) policy. ■ **configuration** shows all of the PBR policies in the virtual router. A PBR policy name can be an alphanumeric string of up to 128 characters in length.

performance

Use the **performance** commands to retrieve performance information for a security device.

You can display information for CPU usage or 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	Displays the current CPU utilization rate for the last minute, the last 5 minutes, and the last fifteen minutes. ■ **detail** displays the CPU utilization for the last 60 seconds, the last 60 minutes, and the last 24 hours.

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 fifteen 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 fifteen 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.

PIM Commands

Use the **pim** context to begin configuring either Protocol Independent Multicast-Sparse Mode (PIM-SM) or Protocol Independent Multicast-Source-Specific Mode (PIM-SSM) for a virtual router.

Context Initiation

Initiating the **pim** 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 **pim** context by executing the **set protocol pim** command.

   device(trust-vr)-> **set protocol pim**

3. Enable PIM (it is disabled by default).

   device(trust-vr/pim)-> **set enable**

4. To exit each context, enter **exit**.

PIM Command List

The following commands are executable in the **pim** context:

accept-group	Use the **accept-group** command to specify the access list that identifies the multicast group(s) for which the virtual router processes PIM messages. Command options: **set, unset**
bsr	Use the **bsr** command to display information about the bootstrap router. Command options: **get**
config	Use the **config** command to display all commands executed to configure the PIM routing instance. Command options: **get**
enable	Use the **enable** command to enable or disable the PIM-SM instance on the virtual router. Command options: **set, unset**
igmp-members	Use the **igmp-members** command to display IGMP membership reports. Command options: **get**

interface	Use the **interface** command to display all interfaces running PIM-SM. Command options: **get**
join-prune	Use the **join-prune** command to display join-prune messages sent to each neighbor. Command options: **get**
mgroup	Use the **mgroup** command to specify from which source(s) and/or RP the multicast group accepts traffic. Command options: **set, unset**
mroute	Use the **mroute** commands to display PIM multicast route table entries. Command options: **get**
neighbor	Use the **neighbor** command to display information about all neighbors discovered for each interface. Command options: **get**
rp	Use the rp command to display the status of the RP (rendezvous point). Command options: **get**
rpf	Use the **rpf** command to display RPF information for a particular source or RP. Command options: **get**
spt-threshold	Use the **spt-threshold** command to specify the data rate in bytes per second that triggers the device to switch from the shared distribution tree to the source-specific distribution tree. Command options: **set, unset**
statistics	Use the **statistics** command to display PIM statics for the virtual router.
zone	Configures the following: ■ an RP candidate in the specified zone ■ a static RP for the specified multicast groups in the named zone Command options: **get, set, unset**

accept-group

Use the **accept-group** command to specify the access list that identifies the multicast group(s) for which the virtual router processes PIM messages.

Before you can execute the **accept-group** command, you must initiate the **pim** context. (See Context Initiation on page 431.)

Syntax

set accept-group *number*

Keywords and Variables

Variable Parameter

set accept-group *number*

number	Specifies the access list that identifies the multicast group(s) for which the virtual router accepts PIM messages.

bsr

Use the **bsr** command to display information about the elected bootstrap router.

Before you can execute the **bsr** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

get bsr

Keywords and Variables

None.

config

Use the **config** command to display all commands executed to configure the PIM routing instance.

Before you can execute the **config** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

get config

Keywords and Variables

None.

enable

Use the **enable** command to enable or disable the PIM-SM instance on the virtual router.

Before you can execute the **enable** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

set enable

Keywords and Variables

None.

igmp-members

Use the **igmp-members** command to display local membership information sent by IGMP.

Before you can execute the **igmp-members** command, you must initiate the **pim** context. (See “Context Initiation” on page 431.)

Syntax

get igmp-members

Keywords and Variables

None.

interface

Use the **interface** command to display all interfaces running PIM-SM.

Before you can execute the **interface** command, you must initiate the **pim** context. (See “Context Initiation” on page 431.)

Syntax

get interface

Keywords and Variables

None.

join-prune

Use the **join-prune** command to display join-prune messages sent to each neighbor.

Before you can execute the **join-prune** command, you must initiate the **pim** context. (See “Context Initiation” on page 431.)

Syntax

get join-prune

Keywords and Variables

None.

mgroup

Use the **mgroup** command to specify from which source(s) and/or RP the multicast group accepts traffic.

Before you can execute the **mgroup** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

set mgroup *mcst_addr* { accept-rp *number* | accept-source *number* }

Keywords and Variables

Variable Parameter

set mgroup *mcst_addr*

ip_addr	Specifies the IP address of the multicast group.

accept-rp

set mgroup *mcst_addr* accept-rp *number*
unset mgroup *mcst_addr* accept-rp

accept-rp	Specifies the access list that identifies the RP(s) from which the device forwards traffic to the multicast group. The device drops traffic for the multicast group if the traffic is from an RP that is not on the specified access list.

accept-source

set mgroup *mcst_addr* accept-source *number*
unset mgroup *mcst_addr* accept-source

accept-source	Specifies the access list that identifies the source(s) from which the device forwards traffic to the multicast group.The device drops traffic for the multicast group if the traffic is from a source that is not on the specified access list.

mroute

Use the **mroute** command to display PIM route-table entries.

Before you can execute the **mroute** command, you must initiate the **pim** context. (See “Context Initiation” on page 431.)

Syntax

get mroute
[
brief |
mgroup *mcst_addr* [detail] [brief] [source *ip_addr* [detail] [brief]]
]

Keywords and Variables

brief

get mroute brief
get mroute mgroup *mcst_addr* brief
get mroute mgroup *mcst_addr* source *ip_addr* brief

brief	Displays summary information about the multicast routes. Displays the source address, multicast group address, and the list of incoming and outgoing interfaces.

detail

get mroute mgroup *mcst_addr* detail
get mroute mgroup *mcst_addr* source *ip_addr* detail

brief	Displays information about the multicast route, including the RPF and type of route. It also provides details on the input and output interfaces.

mgroup

get mroute mgroup *mcst_addr* brief
get mroute mgroup *mcst_addr* detail
get mroute mgroup *mcst_addr* source *ip_addr* [brief | detail]

mgroup	Displays multicast route table entries for the specified multicast group or defines a multicast route for a particular multicast group.

source

get mroute mgroup *ip_addr* source *ip_addr*

source	Specifies the IP address of the source of the multicast traffic.

neighbor

Use the **neighbor** command to display information about all neighbors discovered for each interface.

Before you can execute the **neighbor** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

get neighbor

Keywords and Variables

None.

rp

Use the **rp** command to display the status of the RP (rendezvous point).

Before you can execute the **rp** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

```
get rp
    [
    active |
    all |
    candidate |
    mgroup ip_addr [ active ] |
    proxy
    ]
```

Keywords and Variables

active

get rp active

active	Displays the RP that is actively sending multicast traffic to the multicast groups.

all

get rp all

all	Displays information about all candidate and static RPs. It displays the (*, G) and (S, G) mappings for each RP.

candidate

get rp candidate

candidate	Displays the status of the RP candidates that you configured for each zone on the virtual router.

mgroup

get rp mgroup *ip_addr* [active]

mgroup	Displays information about the group-RP set for the specified multicast group. Specify **active** to display the RP for the specified multicast group.

proxy

get rp proxy

proxy	Displays the proxy-RP status for each zone in the PIM instance of the virtual router.

rpf

Use the **rpf** command to display RPF (reverse path forwarding) information for a particular source or RP.

Before you can execute the **rpf** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

get rpf

Keywords and Variables

None.

spt-threshold

Use the **spt-threshold** command to specify the threshold that triggers the virtual router to switch from the shared distribution tree to the source-based tree.

Before you can execute the **spt-threshold** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

set spt-threshold { *number* | infinity }

Keywords and Variables

Variable Parameter

set spt-threshold *number*

number	Specifies the data rate in bytes per second that triggers the device to switch from the shared distribution tree to the source-specific distribution tree. If you specify **infinity**, the device never switches to a source-specific distribution tree.

zone

Use the **zone** command to configure the following for the specified zone:

- An RP candidate
- A static RP for the specified multicast groups in the named zone
- A proxy-RP

Before you can execute the **zone** command, you must initiate the **pim** context. (See "Context Initiation" on page 431.)

Syntax

get

```
get zone
   [
   zone
      [
   bsr |
   rp { active | all | candidate | mgroup ip_addr [ active ] | proxy }
      ]
   ]
```

set

```
set zone zone rp
   {
   address ip_addr mgroup-list number [ always ] |
   candidate interface interface
      [ mgroup-list number [ holdtime number | priority number ] ] |
   proxy
   }
```

Keywords and Variables

address

set zone *zone* rp address *ip_addr* mgroup-list *number* [always]
unset zone zone rp address *ip_addr*

address	Configures a static RP for the multicast groups specified in the access list. If no group is specified, then this RP is used for any multicast group that has no RP. ■ **zone** *zone* Specifies the zone of the RP. ■ **address** *ip_addr* Specifies the IP address of the RP. This IP address can also be the IP address an interfaces on the device. ■ **mgroup-list** *number* Specifies the access list that identifies the multicast group(s) mapped to the RP. ■ **always** Specifies that this RP should always be used for the specified multicast group even if there is a dynamic group-RP mapping for the same group.

bsr

get zone *zone* bsr

bsr	Displays information about the bootstrap router in the zone.

candidate

set zone *zone* rp candidate interface *interface*
set zone *zone* rp candidate interface *interface* mgroup-list *number* holdtime *number*
set zone *zone* rp candidate interface *interface* mgroup-list *number* priority *number*
unset zone *zone* rp candidate

candidate — Configures an RP candidate in the specified zone.

- **zone** *zone* Specifies the zone of the RP.
- **interface** *interface* Specifies the interface that is advertised as the RP candidate.
- **mgroup-list** *number* Specifies the access list which identifies the multicast group(s) for which the interface is the RP candidate.
- **holdtime** *number* Specifies the holdtime advertised to the bootstrap router.
- **priority** *number* Specifies the priority of the interface as the RP candidate.

When you configure proxy RP, you must configure an RP candidate without a multicast group.

proxy

set zone *zone* rp proxy

proxy	Enables proxy RP in the specified zone.

rp

get zone *zone* rp {...}

rp — Displays information about the RP in the specified zone.

- **active** Displays information about the RP that is sending multicast traffic to the multicast group in the specified zone.
- **all** Displays all RPs, including candidate RPs, in the specified zone.
- **candidate** Displays the configured RP in the zone.
- **mgroup** *ip_addr* Displays the RP for the specifies multicast group.
- **proxy** Displays the proxy-RP for the specified zone.

ping

Use the **ping** commands to check a 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 the **from** keyword description).

Syntax

ping *ip_addr*
[count *number*
[from *interface* | size *number* [time-out *number* | from *interface*]] |
from *interface* |
name-lookup [outgoing-interface]
]

Keywords and Variables

Variable Parameters

ping *ip_addr* [...]

ip_addr	Pings the host at address (*ip_addr*).

Example: The following command pings a host with IP address 172.16.11.2:

ping 172.16.11.2

count

ping *ip_addr* count *number* [...]

count	The ping count (*number*).

from

ping *ip_addr* from *interface*

from	The source interface (*interface*) for an extended ping. For more information on interfaces, see “Interface Names” on page A-I. 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.

Example 1: 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

Example 2: 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

size

ping *ip_addr* count *number* size *number* [...]

size	The packet size (*number*) for each ping.

time-out

ping *ip_addr* 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 three seconds:

ping 10.100.2.11 count 4 size 1000 time-out 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.

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 Certificate Revocation Lists (CRLs)
- Configure PKI-related operations, such as verification of certificate revocation

Syntax

exec

```
exec pki
    {
    convert-cert |
    dsa | rsa
      new-key number [ & ] |
    x509
      {
      install-factory-certs name_str |
      pkcs10 |
      scep
        {
        cert id_num |
        key { id_num | last-key } |
        renew id_num |
      } |
      self-signed-cert key-pair 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 | system } |
      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 id_num* |
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 a virtual system (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. The **&** option directs the device to perform key generation in the background, without waiting for the result. Without this option, the device can wait up to 100 seconds.

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 platform of the security device.

Note: Security appliances generate 1024-bit primes. Security systems generate 1024- and 2048-bit primes. For more information, refer to your product datasheet.

- 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 your product datasheet.

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 **&** option directs the device to perform key generation in the background, without waiting for the result. Without this option, the device can wait 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

self-signed-cert

exec pki x509 self-signed-cert key-pair *id_num*

self-signed-cert	Generates a self-signed certificate using the specified (previously generated) key pair. To learn the ID number for a key pair to use when generating the self-signed certificate, enter the following command: **get pki x509 list key-pair**. The output lists the ID number under the *ID num* heading (not the ID number under *IDX*).

Example: The following command generates a self-signed certificate using the key pair with ID number 70320131:

exec pki x509 self-signed-cert key-pair 70320131

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* | **system** } Displays information on the specified certificate. The keyword **system** refers to the self-signed certificate that the security device automatically generates during bootup.
- **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.
  - **no-preload-ca** Prevents automatic installation of CA certificate (currently a CA certificate from Verisign).
  - **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.
  - **domain-component** *name_str* Sets the domain component value. Devices can use this value in certificates for IPSec logon to VPN gateways. For example, the device could use this as a Group IKE ID, accepting ASN1_DN type IKE identities containing "DC=Engineering, DC=NewYork".
  - **email** *string* Sets the email address.
  - **ip** *ip_addr* Sets the IP 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 certificate authority 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 IP address *ip_addr*.

Example 1: 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

Example 2: The following command refreshes the certificate revocation list daily:

set pki x509 default crl-refresh daily

Example 3: 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 certificate authority 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 the 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"

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 certificate authority, a prompt appears presenting a fingerprint value for the CA certificate. (Otherwise, go on 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 certificate authority 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, 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 Deep Inspection 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 ] ]
    ]
      {
      deny |
      permit |
      reject |
      tunnel { l2tp tunn_str | vpn-group id_num } |
      tunnel vpn tunn_str [ l2tp tunn_str | pair-policy pol_num ]
      }
        [ auth [ server name_str ] | webauth ]
          [
          group-expression string |
          user name_str | user-group name_str
          ]
        ] |
          [ schedule name_str ]
          [ log [ alert ] ]
            [ count [ alarm id_num1 id_num2 ] ]
              [ no-session-backup ]
          [ url-filter ]
                [ traffic ] [ gbw number ]
                  [ priority number ]
                    [ mbw number ] | pbw [ number ]
                      dscp { disable | enable [ value dscp-byte ] }
                    }
```

```
                    ]
                [ infranet-auth [ redirect-all | redirect-unauthenticated ] ] |
                [ attack string
                  { action { close | close-client | close-server |
                      drop | drop-packet | ignore | none } |
                  logging
                  }
                      [ ip-action { block | close | notify }
                        [ target { dst-ip | serv | src-ip | zone |
                          zone-serv } ]
                          [ timeout value ]
                      ]
                ] |
                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 anti-spam name_str
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 policy [ global ] id pol_num disable
set policy [ global ] id pol_num gtp name_str
set policy [ global ] id pol_num idp
```

set (within a policy context)

```
set
    {
    attack string
      { action { close | close-client | close-server | drop | drop-packet | ignore
      | none } |
      logging
      }
        [ ip-action { block | close | notify }
          [ target { dst-ip | serv | src-ip | zone | zone-serv } ]
            [ timeout value ]
        ] |
    av name_str |
    count [ alarm number1 number2 ] |
    di-alert-disable |
    di-severity { info | low | medium | high | critical } |
    dst-address | src-address
      { name_str | negate } |
    idp
    log [ alert | session-init ] |
    name name_str |
    service svc_name |
    src-address { name_str | negate }
    url protocol sc-cpa profile { name_str | ns-profile }
    }
```

Keywords and Variables

all

get policy all

all	Displays information about all security policies.

anti-spam

set policy [global] id *pol_num* anti-spam *name_str*
unset policy [global] id *pol_num* anti-spam

anti-spam	Applies an anti-spam profile to an existing policy.

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 { close | close-client | close-server | drop |
drop-packet | ignore | none }
set policy { … } attack *string* logging
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 one of the following specified actions:

- **action**
  - **close**—logs the event, severs the connection, and sends TCP RST packets to both the client and server.
  - **close client**—logs the event, severs the connection, and sends a TCP RST packet to the client.
  - **close server**—logs the event, severs the connection, and sends a TCP RST to the server.
  - **drop**—logs the event and severs the connection without sending either the client or the server TCP RST packets.
  - **drop packet**—logs the event and drops the packet containing the attack object, but it does not sever the connection.
  - **ignore**—logs the event and stops checking—or ignores—the remainder of the connection.
  - **none**—logs the event but takes no action.
- **logging** By default, the security device logs attacks that it detects through Deep Inspection. To disable logging, enter the policy context and use the command *device(policy:number)->* **unset attack** *string* **logging**.

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 to webserver1 server 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 action close-server**
device(policy:1)-> **set attack HIGH:HTTP:ANOM action close-server**
device(policy:1)-> **set attack HIGH:HTTP:SIGS action close-server**

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

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 FTP, HTTP, POP3, or SMTP traffic to which the policy applies to the specified antivirus (AV) scanner (the internal AV scanner is called “scan-mgr”), which examines the data for viruses. If it finds a virus, the security device drops the data and sends a virus notification message to the client. **Note:** The external antivirus feature is not supported in ScreenOS 5.1.0.

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 the internal AV scanner “scan-mgr”:

set policy id 1 from untrust to dmz r-mail1 mail1 smtp permit av scan-mgr

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.

deny | permit | reject

set policy [global] { ... } permit | deny | reject [...]

deny | permit | reject	■ **deny** blocks the service at the firewall. The security device simply drops the packet. ■ **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”.

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

di-severity (within a policy context)

set di-severity

di-severity	Specifies the severity of events that generate error messages. The possible event levels are **info**, **low**, **medium**, **high**, and **critical**.

disable

set policy [global] id *pol_num* disable

disable	Disables the policy without removing it from the configuration.

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 on zones, see "Zone Names" on page B-I.

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.

gtp

set policy { *pol_num* | id *pol_num* } gtp *name_str*
unset policy { *pol_num* | id *pol_num* } gtp *name_str*

gtp	Identifies the name of the GTP Inspection Object you are assigning to the policy. Before you can assign a GTP Inspection Object to a policy, you must first create the GTP configuration.

id

get policy [global] id *pol_num*
set policy [global] id *pol_num1* { ... }
unset policy id *pol_num* [disable]

id *pol_num*	Specifies a 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

set idp [mode tap]
unset idp [mode]

idp	Enables or disables IDP for the traffic to which the policy applies. By default, IDP is disabled for policies. **mode** Sets or unsets **tap** (passive) mode. By default, IDP is in active mode. In active mode, the security device forwards packets to a security module for inspection. If the security device does not detect an attack, it forwards the packet to its destination. If it does detect an attack, the security device performs an IDP action, such as drop, close-server, close-client, and so on. In **tap** mode, the security device copies packets, forwarding the original packet to its destination and forwarding the copy to a security module for inspection. If the security device detects an attack, it makes an event log entry but does not perform any IDP action.

Example: The following commands create a policy, enter the context of that policy, and then apply IDP in tap mode:

device-> set policy id 1 from trust to untrust any any any permit
device-> set policy id 1
device(policy:1)-> set idp mode tap

infranet-auth

set policy { ... } from *zone1* to *zone2 src_addr dst_addr svc_name* { ... } [...]

infranet-auth	Configures an infranet-auth policy and connects to the Infranet Controller via HTTPS. The **default** infranet-auth policy is for source IP-based enforcement. Before defining this policy, you must create address book entries for the destination and source addresses. There can be multiple of these infranet-auth policies. Infranet-auth policies work with both tunnel traffic and firewall traffic. For user traffic that does not pass through a tunnel, the incoming interface binds to the src-zone. The security device uses the IP address or domain name that you specified when you configured the Infranet Controller instance (refer to "infranet" on page 245) on the security device. ScreenOS 5.4 and higher enables you to configure the captive portal feature, which allows you to automatically *redirect* users to the Infranet Controller or to a preconfigured URL (refer to "infranet" on page 245). (The default infranet-auth policy does not support redirection.) The following two ways of redirection is supported: ■ **redirect-all** Redirects all clear-text traffic to the Infranet Controller or to the URL specified in the Redirect URL field. Use this command if your deployment uses IPSec only. After a user signs in to the Infranet Controller or the specified URL, the Infranet Agent on the client establishes a tunnel between the user and the security device based on the key information received from the Infranet Controller. The security device then applies the VPN policy allowing the encrypted traffic to pass through. **Note:** This option does not allow clear text traffic to pass through the device protecting your network from IP spoofing. ■ **redirect-unauthenticated** Redirects clear-text traffic from unauthenticated users to the Infranet Controller or to the URL specified in the Redirect URL field. Use this command if your deployment uses source-IP only or a combination of source-IP and IPSec. After a user signs into the Infranet Controller or the specified URL, the security device allows the user's clear-text traffic to pass through in source-IP deployments. For IPSec deployments, the Infranet Agent creates a tunnel between the user and the security device. The security device then applies the VPN policy allowing the encrypted traffic to pass through. For more information about deploying an infranet-authentication server, refer to the *Unified Access Control Administration Guide*.

Example 1: Configure a redirect infranet-auth policy for deployments that use source IP only or a combination of source IP and IPSec.

set policy from *source-zone* to *dest-zone src_addr dst_addr* any permit infranet-auth redirect-unauthenticated
set infranet controller name controller1 url "http://10.64.12.1/?target=%dest-url%"

NOTE: In examples 1 and 2, the security device replaces the **?target=%dest-url%** parameter with the protected resource URL and then forwards the protected resource URL in encrypted form to the Infranet Controller.

Example 2: Configure a redirect infranet-auth policy for deployments that use IPSec only.

set policy from *source-zone* to *dest-zone src_addr dst_addr* any permit infranet-auth redirect-all
set infranet controller name controller1 url "http://10.64.12.1/?target=%dest-url%"

Example 3: Configure an infranet-auth policy without redirection.

set policy from *source-zone* to *dest-zone src_addr dst_addr* any permit infranet-auth

ip-action

set policy { ... } permit attack *string* action *string* ip-action *string* [target *string* [timeout *number*]]

ip-action *string* — Activates additional brute-force attack defenses to Deep Inspection (DI) detection. A brute-force attack occurs when an attacker barrages a target with every possible combination of attacks until one succeeds. Attackers might use brute-force attacks when attempting to log in, discover protected resources, or break encryption keys. If the security device detects a brute-force attack, it applies the specified IP action for a certain period to other packets with a set of elements that match a defined target.

- **ip-action** Specifies one of the following actions that the security device performs when it detects a brute-force attack:
  - **block** The security device logs the event and drops all further traffic matching the target definition for the period specified in the timeout setting.
  - **close** The security device logs the event and drops all further traffic matching the target definition for the period specified in the timeout setting, then sends a Reset (RST) for TCP traffic to the source and destination addresses.
  - **notify** The security device logs the event but does not take any action against further traffic matching the target definition for the period specified in the timeout setting.

- **target** Specifies a set of elements that must match for the security device to consider a packet to be part of a brute-force attack. The specified set of elements in an IP packet arriving during a specified timeout period must match that in the packet that the security device detected as part of a brute-force attack in order for the subsequent packet to be considered part of the same attack. The default is **serv**.
  - **dst-ip** The destination IP address
  - **serv** The source and destination IP addresses, destination port number, and protocol
  - **src-ip** The source IP address
  - **zone** The security zone to which the ingress interface is bound; that is, the source security zone from which the attacking packets originate
  - **zone-serv** The source security zone, source and destination IP addresses, destination port number, and protocol
- **timeout** A period following brute-force attack detection during which the security device performs an IP action on packets matching specified target parameters. The default is 60 seconds.

Example: The following command applies Deep Inspection to HTTP traffic from any host in the Untrust zone to an HTTP server ("hpp1") in the DMZ. It searches for attacks in the attack group "HIGH:HTTP:ANOM", which contains two brute force attack objects. If the security device detects any attack included in that group, it drops the traffic and sends a TCP RST to the webserver. If the security device detects either of the two brute force attacks, it also drops further HTTP traffic (using TCP to port 80) to that server from any host in the Untrust zone for the next 30 seconds:

set policy from untrust to dmz any http1 http permit attack HIGH:HTTP:ANOM action close ip-action close target zone-serv timeout 30

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] [session-init] { ... }

log	Enables logging when a session ends.
alert	Enables the syslog alert feature.
session-init	Enables logging when a starts.

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. The security device generates logs when sessions end.
- 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 using the egress interface IP address (in which case, you do not specify a DIP pool) or with addresses from a Dynamic IP (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.

Example 1: 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

Example 2: 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

Example 3: 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-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

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*] | pbw [*number*]
  dscp { disable | enable [value *dscp-byte*] }

traffic gbw

Defines the guaranteed bandwidth 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 in kilobits per second. Traffic beyond this limit is throttled and dropped.
- **pbw** *number* Defines the policing bandwidth in kilobits per second on the ingress side of the security device. Traffic beyond this limit is dropped.
- **dscp { enable [value** *dscp-byte* **]** | **disable }** Enables or disables a mapping of the eight ScreenOS priority levels to the Differentiated Services—DiffServ—Codepoint (DSCP) marking system, or optionally specifies a DSCP value independent of a ScreenOS priority setting..

  In the ScreenOS system, 0 is the highest priority and seven is the lowest. When you enable DSCP and do not specify a value, ScreenOS overwrites the first 3 bits in the DiffServ field (see RFC 2474), or the IP precedence field in the TOS byte (see RFC 1349), in the IP packet header. When you set a dscp-byte value (0-63), ScreenOS overwrites the first 6 bits of the TOS field to specify the class or type of network service.

Example: The following command:

- 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

url

set url protocol sc-cpa profile { *name_str* | ns-profile }

profile	Specifies the URL filtering profile that you are binding to the specified policy. Only one URL profile can be linked to a policy. Use this command when configuring the integrated URL filtering feature. For information about this feature, refer to the *Concepts & Examples ScreenOS Reference Guide*.

url-filter

set policy { ... } url-filter

url-filter	Enables URL filtering on the security device.

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.)

CAUTION: Setting the port mode removes any existing configurations on the device and requires a system reset.

Syntax

exec port-mode { combined | dmz-dual-untrust | dual-untrust | dual-dmz | extended |
home_work | trust-untrust }

Keywords and Variables

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.

dmz-dual-untrust

Defines the following port, interface, and zone bindings:

- Binds the Ethernet ports 1 and 2 to the ethernet1 interface, which is bound to the Trust security zone.
- Binds the Ethernet port 3 to the ethernet2 interface, which is bound to the DMZ security zone.
- Binds the Ethernet port 4 to the ethernet3 interface, which is bound to the Untrust security zone.
- Binds the Untrust Ethernet port to the ethernet4 interface, which is bound to the Untrust security 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.

dual-dmz Defines the following port, interface, and zone bindings:

- Binds the Ethernet port 1 to the ethernet1 interface, which is bound to the Trust security zone.
- Binds the Ethernet port 2 to the ethernet2 interface, which is bound to the DMZ security zone.
- Binds the Ethernet port 3 to the ethernet3 interface, which is bound to the DMZ2 security zone.
- Binds the Ethernet port 4 to the ethernet4 interface, which is bound to the Untrust security zone.
- Binds the Untrust Ethernet port to the ethernet5 interface, which is bound to the Untrust security zone.

extended Defines the following port, interface, and zone bindings:

- Binds the Ethernet ports 1 and 2 to the ethernet1 interface, which is bound to the Trust security zone.
- Binds the Ethernet ports 3 and 4 to the ethernet2 interface, which is bound to the DMZ security zone.
- Binds the Untrusted Ethernet port to the ethernet3 interface, which is bound to the Untrust security zone.
- Binds the Modem port to the serial interface, which you can bind as a backup interface to the Untrust security zone.

home-work Defines the following port, interface, and zone bindings:

- Binds the Untrusted Ethernet port to the ethernet3 interface, which is bound to the Untrust 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.

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.

NOTE: Setting the port mode removes any existing configurations on the device and requires a system reset.

ppp

Point-to-Point Protocol (PPP) provides a standard method for encapsulating Network Layer protocol information over point-to-point links. PPP encapsulation is defined in RFC 1661, *The Point-to-Point Protocol (PPP)*.

Use the **ppp** commands to configure PPP or to display current PPP configuration parameters.

Syntax

clear

clear ppp [name *profile_name*]

get

get ppp profile { *profile_name* | all }

set

set ppp profile *profile_name*
    {
        auth
            {
            local-name *name_str*
            secret *string*
            type { any | chap | none | pap }
            } |
        netmask *netmask* |
        passive |
        static-ip
    }

Keywords and Variables

profile

set ppp profile *profile_name* { ... }
get ppp profile { all | *profile_name* }

profile	Creates and configures a PPP access profile, which specifies authentication parameters for the PPP link. You bind the PPP access profile to an interface with the **set interface** command. You can configure the following parameters in the PPP access profile:

- **auth** Specifies the authentication method to be used when establishing the link or the hostname to be used in Challenge Handshake Authentication Protocol (CHAP) requests and responses. Specify one of the following values:
  - **local-name** *name_str* Specifies the local client name.
  - **secret** *string* Specifies the local client secret.
  - **type** Specifies the PPP authentication type:

  any Sets the PPP profile to negotiate any type of the supported PPP authentication.

  chap Sets the PPP profile to use Challenge Handshake Authentication Protocol.

  none Sets the PPP profile to not use any authentication type.

  pap Sets the PPP profile to use Password Authentication Protocol.
- **netmask** Specifies the netmask for the interface. The default is 255.255.255.255.
- **passive** Directs the interface not to challenge its peer and to respond only when challenged. This is disabled by default.
- **static-ip** Directs the interface to use an IP address that you have manually configured for the interface.

For the **get** command, you can show information either for a specified profile or for all configured profiles.

The **clear** command allows you to clear all parameters for a specified profile.

pppoa

Use the **pppoa** commands to configure PPPoA or to display current PPPoA configuration parameters.

Point-to-Point Protocol over ATM (PPPoA) is usually used for PPP sessions that are to be terminated on a security device with an ADSL interface. PPPoA is primarily used for business class services because it does not require a desktop client (which is required for PPPoE termination).

Syntax

clear

clear [cluster] pppoa [name *name_str*]

exec

exec pppoa [name *name_str*] { connect | disconnect }

get

get pppoa { all | name *name_str* }

set

```
set pppoa [ name name_str ]
    {
    authentication { CHAP | PAP | any } |
    auto-connect number |
    clear-on-disconnect |
    idle-interval number |
    interface [ interface ] |
    netmask [ mask ] |
    ppp
      {
      lcp-echo-retries number |
      lcp-echo-timeout number
      } |
    static-ip |
    update-dhcpserver |
    username name_str password pswd_str
    }
```

Keywords and Variables

all

get pppoa all

all	Displays information for all PPPoA instances.

authentication

set pppoa authentication { CHAP | PAP | any }
unset pppoa authentication { CHAP | PAP }

authentication	Sets the authentication methods to **CHAP**, **PAP**, or **any**. (The **any** option gives preference to CHAP.) The default authentication is **any** (both CHAP and PAP). To set authentication to CHAP only, first execute **unset pppoa authentication PAP**.

auto-connect

set pppoa auto-connect *number*
unset pppoa 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.) This is disabled by default.

clear-on-disconnect

set pppoa [name *name_str*] clear-on-disconnect
unset pppoa clear-on-disconnect

clear-on-disconnect	Directs the security device to clear the IP address and the gateway for the interface once PPPoA disconnects. By default, this is disabled; that is, the IP address and gateway for the interface remain when PPPoA disconnects. If you do not specify **name**, ScreenOS sets the parameter for the default instance untrust.

connect | disconnect

exec pppoa [name *name_str*] { connect | disconnect }

connect	Starts a PPPoA connection for an instance. (Each instance can be bound to an interface.)
disconnect	Takes down a PPPoA connection.

idle-interval

set pppoa idle-interval *number*
unset pppoa idle-interval

idle-interval	Sets the idle timeout, which is time elapsed (in minutes) before the security device terminates a PPPoA connection due to inactivity. Valid range is 0-10000 minutes. Specifying 0 turns off the idle timeout and the device never terminates the connection. The default is 30 minutes.

interface

set pppoa interface [*name_str*]
unset pppoa interface

interface	Specifies the ADSL interface for PPPoA encapsulation.

name

exec pppoa [name *name_str*] { connect | disconnect }
get pppoa [name *name_str* | all]
set pppoa [name *name_str*] ...
unset pppoa [name *name_str*]

name	Specifies or defines the name for a specific PPPoA instance. You can assign a username and password, an interface, and other PPP/PPPoA 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 PPPoA instance.

- Username user1 and password 123456
- PPPoA instance pppoa-user-1 bound to the ethernet2 interface

set pppoa name pppoa-user-1 username user1 password 123456
set pppoa name pppoa-user-1 interface ethernet2

netmask

set pppoa netmask *mask*
unset pppoa netmask

netmask	Specifies a PPPoa subnet mask that the device assigns to the interface bound to the PPPoA instance (after establishment of the connection). The default netmask is 255.255.255.0. 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 pppoa ppp { ... }
unset pppoa ppp { ... }

ppp	Specifies PPP parameters. ■ **lcp-echo-retries** the number of unacknowledged LCP Echo requests before connection is terminated. Valid range is 1-30. The default is 10. ■ **lcp-echo-timeout** the time that elapses between transmission of two LCP Echo requests. Valid range is 1-1000 seconds. The default is 180 seconds.

static-ip

set pppoa static-ip
unset pppoa static-ip

static-ip	Specifies that your connection uses the static IP address assigned to your device's interface. This is disabled by default.

update-dhcpserver

set pppoa update-dhcpserver
unset pppoa update-dhcpserver

update-dhcpserver	Specifies that the DHCP server (on the device) automatically updates DNS parameters received through the PPPoA connection. This is enabled by default.

username

set pppoa username *name_str* password *pswd_str*

username	Sets the user name and password for authentication.

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.

- User name *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, DHCP, 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 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 user name 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*

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
```

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** commands to restart the security 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.

RIP Commands

Use the **rip** context to begin configuring Routing Information Protocol (RIP) for a virtual router.

CAUTION: RIP is *not* supported over unnumbered tunnel interfaces. All interfaces that use RIP must be numbered. Any attempt to configure and run an unnumbered interface using RIP can lead to an unpredictable routing failure.

Context Initiation

Initiating the **rip** 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 **rip** context by executing the **set protocol rip** command.

   device(trust-vr)-> **set protocol rip**

3. Enable RIP (it is disabled by default).

   device(trust-vr/rip)-> **set enable**

RIP Command List

The following commands are executable in the **rip** context. Click on a keyword in the table to go to complete syntax and usage information.

advertise-def-route	Use the **advertise-def-route** commands to advertise the default route (0.0.0.0/0) of the current virtual router in the RIP routing domain. Every virtual router can 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**
alt-route	Use the **alt-route** commands to set the maximum number of alternate routes in the RIP database for a network prefix. Command options: **set, unset**
config	Use the **config** command to display all commands executed to configure the RIP routing instance. Command options: **get**

database	Use the **database** command to display the RIP database in the virtual router. Command options: **get**
default-metric	Use the **default-metric** commands to set the RIP metric for redistributed routes. The default value is 10. Command options: **set, unset**
enable	Use the **enable** commands to enable or disable RIP in the virtual router. Command options: **set, unset**
flush-timer	Use the **flush-timer** commands to configure the number of seconds that elapse before the virtual router automatically removes an invalidated route. The default is 120 seconds. Command options: **set, unset**
garbage-list	Displays all routes currently contained in the RIP 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 options: **get**
hold-timer	Use the **hold-timer** commands to configure the number of seconds that elapse before the virtual router updates the routing table when RIP detects a route with a high metric. Command options: **set, unset**
interface	Use the **interface** command to display all RIP 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 RIP neighbors allowed. The default is 16. Command options: **set, unset**
neighbors	Use the **neighbors** command to display the status of RIP neighbors. Command options: **get**
no-source-validation	Use the **no-source-validation** commands to accept responses from RIP neighbors in other subnets or to reject such responses. The default action is to reject the responses. Command options: **set, unset**
poll-timer	Use the **poll-timer** commands to set the interval and number of times that triggered requests are sent over the demand circuit to check if the other end of the demand circuit has come up. Command options: **set, unset**

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 an interface with an IP address becoming available
- Routes imported from other virtual routes

Command options: **set, unset**

reject-default-route

Use the **reject-default-route** commands to cause RIP to reject a default route learned from a neighbor.

Command options: **get, set, unset**

retransmit-timer

Use the **retransmit-timer** commands to set the interval and number of times that triggered messages waiting for acknowledgement or a response are retransmitted over the demand circuit.

Command options: **set, unset**

route-map

Use the **route-map** commands to filter routes and offset the metric to a RIP 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**

summary

Use the summary command to display summary routes.

Command options: **get**

summary-ip

Use the **summary-ip** command to create a summary route that corresponds to a summary range.

Command options: **set, unset**

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 RIP timers.

Command options: **get**

trusted-neighbors

Use the **trusted-neighbors** commands to set an access list that defines RIP 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 RIP 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**

version	Use the **version** command to set the RIP protocol version for the virtual router. Command options: **set, unset**

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 default route is a non-RIP route.

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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

get

get advertise-def-route

set

set advertise-def-route [always] { metric *number* | preserve-metric }

Keywords and Variables

always

set advertise-def-route always ...

always	Directs the routing instance to advertise the non-RIP 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, which is a route redistributed from a protocol other than RIP. 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.

alt-route

Use the **alt-route** commands to set the maximum number of alternate routes that the security device maintains in the RIP database for a network prefix.

Before you can execute the **alt-route** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set alt-route *number*

Keywords and Variables

number

set alt-route *number*

number	Sets the maximum number of alternate routes in the RIP database for a network prefix. Enter a value between 0 and 3. The default value is 0, which means that there are no alternate routes in the database for a network prefix.

config

Use the **config** command to display all commands executed to configure the RIP local virtual router.

Before you can execute the **config** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

get config

Keywords and Variables

None.

database

Use the **database** command to display the RIP database in the local virtual router.

Before you can execute the **database** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

get database [prefix *ip_addr/mask*]

Keywords and Variables

prefix

get database prefix *ip_addr/mask*

prefix	Shows specific RIP entries in detail.

default-metric

Use the **default-metric** commands to set the RIP metric for redistributed routes.

Before you can execute the **default-metric** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set default-metric *number*

Keywords and Variables

Variable Parameter

set default-metric *number*

number	The metric for the routes redistributed into RIP. This metric value can be from 1 to 15.

enable

Use the **enable** commands to enable or disable RIP from the current virtual router.

Before you can execute the **enable** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

set flush-timer *number*

Keywords and Variables

Variable Parameter

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 RIP 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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

get garbage-list

Keywords and Variables

None.

hold-timer

Use the **hold-timer** commands to configure the time that elapses before the virtual router makes any updates into the routing table whenever RIP detects unreachable routes and higher metric routes. This minimizes the effects of route flapping to the routing table.

Before you can execute the **hold-timer** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set hold-timer *number*

Keywords and Variables

Variable Parameter

set hold-timer *number*

number	The number of seconds that elapses before the virtual router updates the routing table when RIP detects a route with a high metric. The minimum value should be three times the **update-timer** value. The sum of the **update-timer** and the **hold-timer** values should not exceed the **flush-timer** value. The default value is 90 seconds.

interface

Use the **interface** command to display all RIP interfaces on the current virtual router.

Before you can execute the **interface** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

set invalid-timer *number*

Keywords and Variables

Variable Parameter

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 RIP neighbors, which belong to the specified virtual router, allowed on an interface.

Before you can execute the **max-neighbor-count** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set max-neighbor-count *number*

Keywords and Variables

Variable Parameter

set max-neighbor-count *number*

number	The maximum number of RIP neighbors allowed. This value can be from one to the maximum value possible for your security device. The default is platform-dependent. Refer to the datasheet for the maximum limit for a particular device.

neighbors

Use the **neighbors** command to display the status of all RIP neighbors.

Before you can execute the **neighbors** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

get neighbors

Keywords and Variables

None.

no-source-validation

Use the **no-source-validation** commands to accept responses from RIP neighbors in different subnets. If you do not set this switch, the virtual router does not process responses from neighbors in other subnets.

Before you can execute the **no-source-validation** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set no-source-validation

Keywords and Variables

None.

poll-timer

Use the **poll-timer** commands to configure the interval at which triggered requests are sent over a demand circuit to check if the other end of the circuit has come up.

Before you can execute the **poll-timer** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set poll-timer *number* [retry-count *number*]

Keywords and Variables

Variable Parameter

set poll-timer *number*

number	The interval, in number of seconds, at which triggered requests are sent over the demand circuit to check if the other end of the circuit has come up. The default value is 180 seconds (3 minutes).

retry-count

set poll-timer *number* retry-count *number*

retry-count	The number of times that the triggered requests are sent before the demand circuit is declared to be down. The default is 0, which means that the triggered requests are sent indefinitely.

redistribute

Use the **redistribute** commands to import known routes from a router running a different protocol into the current RIP 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 **rip** context. (See “Context Initiation” on page 491.)

Syntax

get

get routes-redistribute
get rules-redistribute

set

set redistribute route-map *name_str* protocol
    { bgp | connected | discovered | 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. ■ **discovered** specifies that the route map performs an action only on discovered routes in the subnetwork. ■ **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 RIP routing instance:

device(trust-vr/rip)-> **set redistribute route-map map1 protocol bgp**

reject-default-route

Use the **reject-default-route** commands to cause RIP to reject default routes learned from a neighbor in the RIP domain.

Before you can execute the **reject-default-route** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

get

get reject-default-route

set

set reject-default-route

Keywords and Variables

None.

retransmit-timer

Use the **retransmit-timer** command to configure the interval at which triggered responses are retransmitted over a demand circuit.

Before you can execute the **retransmit-timer** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set retransmit-timer *number* [retry-count *number*]

Keywords and Variables

Variable Parameter

set retransmit-timer *number*

number	The interval, in number of seconds, at which triggered responses are retransmitted over a demand circuit. The default is 5 seconds.

retry-count

set retransmit-timer *number* retry-count *number*

retry-count	The number of times any response is retransmitted before the demand circuit is placed into POLL state. The default is 10 times.

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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

get

get route-map

set

set route-map *name_str* { in | out }

Keywords and Variables

Variable Parameter

set route-map *name_str*

name_str	The name of the route map to filter routes.

in

set route-map *name_str* in
unset route-map *name_str* in

in	Specifies the route map is applied to routes to be learned by RIP.

out

set route-map *name_str* out
unset route-map *name_str* out

out	Specifies the route map is applied to routes to be advertised by RIP.

Example: The following command applies the route map map1 to routes to be advertised by RIP:

device(trust-vr/rip)-> **set route-map map1 out**

routes-redistribute

Use the **routes-redistribute** command to display details about routes imported from other protocols into RIP.

Before you can execute the **routes-redistribute** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

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 RIP.

Before you can execute the **rules-redistribute** command, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

get rules-redistribute

Keywords and Variables

None.

summary

Use the **summary** command to display summary routes configured with the **summary-ip** command.

Before you can execute the **summary** command, you must initiate the **rip** context. (See “Context Initiation” on page 491.)

Syntax

get summary

Keywords and Variables

None.

summary-ip

Use the **summary-ip** commands to summarize the routes that are advertised by RIP. You enable the advertising of summary routes on a per-interface basis.

Before you can execute the **summary-ip** commands, you must initiate the **rip** context. (See “Context Initiation” on page 491.)

Syntax

set summary-ip *ip_addr/mask* [metric *number*]

Keywords and Variables

Variable Parameter

set summary-ip *ip_addr/mask*
unset summary-ip *ip_addr/mask*

ip_addr/mask	The summary range that encompasses constituent routes.

metric

set summary-ip *ip_addr/mask* [metric *number*]

metric	Specifies the metric for the summary route. If no metric is specified, the largest metric for a constituent route is used.

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 **rip** context. (See “Context Initiation” on page 491.)

Syntax

set threshold-update *number*

Keywords and Variables

Variable Parameter

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 RIP timers.

Before you can execute the **timer** command, you must initiate the **rip** context. (See “Context Initiation” on page 491.)

Syntax

get timer

Keywords and Variables

None.

trusted-neighbors

Use the **trusted-neighbors** commands to specify an access list that defines allowed RIP neighbors.

Before you can execute the **trusted-neighbors** commands, you must initiate the **rip** context. (See “Context Initiation” on page 491.)

Syntax

get

get trusted-neighbors

set

set trusted-neighbors *id_num*

Keywords and Variables

Variable Parameter

set trusted-neighbors *id_num*

id_num	The number of the access list that defines the allowed RIP neighbors.

update-timer

Use the **update-timer** commands to set the interval that RIP sends route updates to neighbors.

Before you can execute the **update-timer** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set update-timer *number*

Keywords and Variables

Variable Parameter

set update-timer *number*

number	The interval, in seconds, that RIP 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 **rip** context. (See "Context Initiation" on page 491.)

Syntax

get update-threshold

Keywords and Variables

None.

version

Use the **version** commands to set the RIP protocol version in the virtual router.

Before you can execute the **version** commands, you must initiate the **rip** context. (See "Context Initiation" on page 491.)

Syntax

set version { v1 | v2 }

Keywords and Variables

v1 | v2

set version v1 | v2

v1 | v2	Sets the RIP protocol version in the virtual router and on all RIP interfaces to either version 1 or version 2. The default is version 2. You can override the protocol version on a per-interface basis.

route

Use the **route** commands to display entries in the static route table.

The **get route** command displays the following:

- The IP address, netmask, interface, gateway, protocol, preference, metric, and owner vsys

  The value of **protocol** can be any of the following:

  - **C** (Connected)
  - **S** (Static)
  - **A** (Auto Exported)
  - **D** (Auto Discovered)
  - **I** (Imported from another virtual router)
  - **iB** (internal BGP)
  - **eB** (external BGP)
  - **H** (Host)
  - **O** (OSPF)
  - **P** (Permanent)
  - **R** (RIP)
  - **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

get route
[
id *id_num* |
ip [*ip_addr*] |
prefix *ip_addr/mask* |
protocol { bgp | connected | discovered | imported | ospf | rip | static } |
source [id *number* | in-interface | ip *ip_addr* | prefix *ip_addr/mask*] |
summary
]

Keywords and Variables

id

get route id *id_num*

id	Displays a specific route for the ID number *id_num*.

Example: The following command displays the route information for a route with ID number 477:

get route id 477

in-interface

get route source in-interface

in-interface	Displays Source Interface-Based Routes (SIBR) routes.

ip

get route ip *ip_addr*

ip	Displays a specific route for the target IP address (*ip_addr*).

Example: The following command displays the route information to a machine with the IP address 172.16.60.1:

get route ip 172.16.60.1

prefix

get route prefix *ip_addr/mask*

prefix	Displays routes within a specified subnet (*ip_addr/mask*).

Example: The following command displays the routes within the subnet 1.1.1.1/24:

get route prefix 1.1.1.1/24

protocol

get route protocol { bgp | connected | discovered | imported | ospf | rip | 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 discovered routes.
- **imported** Directs the device to display imported routes.
- **rip** Directs the device to display RIP routes.
- **ospf** Directs the device to display only OSPF routes.
- **static** Directs the device to display only static routes.

source

get route source [id *number* | in-interface | ip *ip_addr* | prefix *ip_addr/mask*]

source — Displays source routes.

- **id** *number* shows a particular source route.
- **in-interface** shows source interface-based routes (SIBR).
- **ip** *ip_addr* shows a route for a particular IP address.
- **prefix** *ip_addr/mask* shows routes within a subnet.

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 ] |
    stat
    ]
```

Keywords and Variables

Variable Parameter

clear [cluster] sa *id_num*

id_num	Specifies a security association (SA) ID number.

active

get sa active [stat]

active	Displays the active SA(s).

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 an SA entry for the specified ID number (id_num).

inactive

get sa inactive [stat]

inactive	Displays the inactive SA(s).

stat

get sa [active | inactive] stat

stat

Shows the SA statistics for the device. Also displays active or inactive SA statistics.

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** commands to debug messages for each Security Association (SA) filter.

Syntax

get

get sa-filter

set

set sa-filter *ip_addr*

unset

unset sa-filter { *ip_addr* | all }

Keywords and Variables

Variable Parameter

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

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	Clears the statistics for a particular SA (id_num).

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 Internet Key Exchange virtual private network (IKE VPN) tunnel.

Syntax

clear

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	Clears the statistics for a particular SA (id_num).

save

Use the **save** commands to save ScreenOS images to a security device and to save device configuration settings to or from a security device. You can also use this command to save the authentication certificate to the security device for authenticating ScreenOS images and attack object database downloads for Deep Inspection (DI).

Syntax

save

```
save
    [
    attack-db from tftp ip_addr filename to flash [ from interface ] |
    config
      [
      all-virtual-system |
      to { flash | slot1 filename | tftp ip_addr filename | usb 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 ] |
          usb filename |
        }
      ]
    image-key { tftp ip_addr filename [ from interface ] | usb filename } |
    software from
      {
      flash |
      slot1 filename |
      tftp ip_addr filename |
      usb filename
      }
      to
        {
        flash |
        slot1 filename |
        tftp ip_addr filename |
        usb filename
        }
          [ from interface ]
```

Keywords and Variables

all-virtual-system

save config all-virtual-system

all-virtual-system	Saves all virtual system configurations.

attack-db

save attack-db from tftp *ip_addr filename* to flash [from *interface*]

attack-db	Saves the attack database to the security device.

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: 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

from { ... } to

save config from { ... } to { ... }
save software from { ... } to { ... }

from	Saves from the specified source.
to	Saves to the specified destination.

Example: 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

image-key

save image-key from tftp *ip_addr filename*

image-key	Saves the authentication certificate (imagekey.cer) to the security device. After you save this certificate onto the security device, the device uses it to verify the integrity of ScreenOS images when you save them to the device and when it reboots. The security device also uses this certificate to verify the integrity of Deep Inspection (DI) attack object database files during the download process. The authentication certificate is available on the Documentation CD-ROM that ships with each security device. It is also available online at the Juniper Networks Web site. Log in at www.juniper.net/support/, click **ScreenOS Software** in the Download Software section, and click **Download the Authentication Certificate** at the top of the page. Saving this certificate onto a security device automatically causes the device to perform authentication checks on ScreenOS images and DI attack object database downloads. To stop these checks, you must remove the authentication certificate, using the **delete crypto auth-key** command.

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

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*
save software from tftp *filename* to { ... } [from *interface*]

tftp	Saves from (or to) a file on a TFTP server.

Example: The following command saves an authentication certificate onto a security device from a file named imagekey.cer on a TFTP server at *10.10.1.2*:

save image-key tftp 10.10.1.2 nskey.cer

usb

save config from usb *filename* to { ... }
save image-key usb *filename*
save software from usb *filename* to { ... }

usb	Saves from (or to) a file on a USB key using the USB host module.

Example: The following command saves the file named nskey.txt to the USB storage device:

save image-key usb nskey.txt

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
    [
    svc_name |
    group [ name_str ] |
    pre-defined |
    timeout { other | tcp | udp } [ port number1 [ number2 ] ] |
    user
    ]
```

set

```
set service svc_name
    {
    +
      {
      icmp type number code number |
      ptcl_num | tcp | udp
        { src-port number-number dst-port number-number }
      ms-rpc uuid UUID_string |
      sun-rpc { program id_num1-id_num2 }
      } |
    protocol
      {
        { ptcl_num | tcp | udp }
          [ src-port number-number ]
            [ dst-port number-number [ timeout { number | never } ] ]
        icmp type number code number |
        ms-rpc { uuid UUID_string [ timeout { number | never } ] } |
        sun-rpc
          { program id_num1-id_num2}
            [ timeout { number | never } ]
      } |
    timeout { number | never }
    }
```

Keywords and Variables

Variable Parameters

get service *svc_name*
set service *svc_name* [...]
unset service *svc_name*

svc_name	Identifies a service by name.

+

set service *svc_name* + { ... }

+	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 *svc_name* 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.
- **ms-rpc** specifies a Microsoft RPC service.
  - **uuid** specifies the interface (16 bytes).
- **sun-rpc** specifies a Sun RPC service
  - **program** specifies the program (32 bit integer).
- **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 *svc_name* 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 *svc_name* timeout { *number* | never }
unset service *svc_name* timeout

timeout	Sets or displays the timeout value for sessions created on a port for TCP, UDP, or other protocols. You can specify session timeout value (*number*) in minutes or as **never**.

Example 1: The following command is a service named *telnet* with a timeout value of *10* minutes:

set service telnet timeout 10
Example 2: 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 maximum 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 platform. (For example, on a 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* |
ike-nat |
rm |
service name_*str* |[tunnel]
[hardware [0 | 1 | 2 | 3 | 4 | 5 |] |
[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 *number*] [hardware] [0 | 1 | 2 | 3 | 4 | 5 |]
]

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

ike-nat

get session ike-nat

ike-nat	Identifies all IKE NAT ALG session information.

hardware

get session [hardware] [0 | 1 | 2 | 3 | 4 | 5 |]

hardware	Displays session information on hardware acceleration chip. ■ 0—Shows asic 0 sessions. ■ 1—Shows asic 1sessions. ■ 2—Shows asic 2 sessions. ■ 3—Shows asic 3 sessions. ■ 4—Shows asic 4 sessions. ■ 5—Shows asic 5 sessions.

rm

get session rm

rm	Displays sessions for resource management.

service

get session service *name string*

service	Displays sessions for specific service or service group defined by the **set service** command.

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.
dst-ip *ip_addr*	Identifies all sessions initiated by packets containing destination IP address *ip_addr*.

Example: 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

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*
clear [cluster] session [...]
get session [...] vsd-id *id_num* [hardware] [0 | 1 | 2 | 3 | 4 | 5 |]

vsd-id *id_num* — Identifies all sessions that belong the VSD group *id_num*. The keyword **hardware** displays hardware sessions and, optionally, information about sessions on specific hardware acceleration chips, as follows:

- 0—Shows asic 0 sessions.
- 1—Shows asic 1 sessions.
- 2—Shows asic 2 sessions.
- 3—Shows asic 3 sessions.
- 4—Shows asic 4 sessions.
- 5—Shows asic 5 sessions.

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

sip

Use the **sip** commands to configure SIP functionality on the security device and also to obtain information.

Syntax

get

```
get sip
    {
    call |
    setting
    }
```

set

set sip
{
media-inactivity-timeout *number* |
protect deny [dst-ip *ip_addr/mask* | timeout [*number*]] |
signaling-inactivity-timeout *number*
}

Keywords and Variables

call

get sip call

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 specifications sheet for your product.

media-inactivity-timeout

set sip media-inactivity-timeout *number*
unset sip media-inactivity-timeout

media-inactivity-timeout	Configures or removes the maximum length of time (in seconds) a call can remain active without any media (RTP or RTCP) traffic within a group. Each time a RTP or RTCP packet occurs within a call, this timeout resets. The default setting is 120 seconds.

protect deny

set sip protect deny [dst-ip *ip_addr/mask* | timeout [*number*]]
unset sip protect deny [dst-ip *ip_addr/mask* | timeout]

protect deny	Specifies that repeat SIP INVITE requests from a source be denied to a proxy server that denied the initial request. ■ **dst-ip** *ip_addr/mask* specifies the address of the proxy server. ■ **timeout** [*number*] specifies the number of seconds (the default is 3) the proxy server will deny repeated SIP INVITE requests before it begins accepting them again.

setting

get sip setting

setting	Displays the inactivity timeout parameters of the SIP ALG (application-layer gateway) for SIP signaling and SIP media, and the destination address of a SIP proxy server protected from repeat SIP INVITE requests the proxy server initially rejected.

signaling-inactivity-timeout

set sip signaling-inactivity-timeout *number*
unset sip signaling-inactivity-timeout

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).

sm-ctx

Use the **sm-ctx** commands to view the status of security modules (SM) on your security device.

Syntax

get

get sm-ctx { pkt | status }

Keywords and Variables

sm-ctx

get sm-ctx status

pkt	Displays security module's packet counts in the following four columns of output: ■ **SM**—Security module number. ■ **TX**—Packet number sent to the security module (16 bits counter). ■ **RX**—Packet number received from the security module (16 bits counter). ■ **SN**—Security module's engine start number. Typically, it is **1** (initial start). Each time you restart the engine restart (crash), this counter is incremented by 1.
sm-ctx status	Displays information about the security modules in your security device in the following four columns of output: ■ **SM CPU**—Displays the CPU numbers for each security module. CPU 1 and 2 are in security module 1, CPU 3 and 4 are in security module 2, and CPU 5 and 6 are in security module 3. ■ **aval**—If a security module is functioning properly, **1** appears in this column. If a security module does not occupy one of the security module slots or if it is malfunctioning, column 2 shows **0**. ■ **ena**—Always shows the number 1. ■ **Sess_cnt**—Lists the number of sessions running on the CPUs on each security module.

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

clear

clear snmp statistics

get

get snmp [auth-trap | community *name_str* | settings | statistics]

set

set snmp
  {
  auth-trap enable |
  community *name_str*
    { read-only | read-write }
      [
      trap-off] |
      trap-on [traffic] |
      version { any | v1 | v2 }
      ] |
  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.

Example 1: 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

Example 2: 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 a 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 *juniper*.

- 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 juniper read-write trap-on
set snmp host juniper 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.

statistics

clear snmp statistics
get snmp statistics

statistics	Displays or clears SNMP statistics.

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 on 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

clear

clear socket id *id_num*

get

get socket [id *id_num*]

Keywords and Variables

id

clear socket id *id_num*
get socket id *id_num*

id	Clears or displays the information for an identified socket (*id_num*).

Example: The following command displays the information concerning socket 5:

get socket id 5

ssh

Use the **ssh** commands to configure the Secure Shell (SSH) server task.

The SSH server task is an SSH-compatible server application that resides on the security device. When you enable the SSH server task, SSH client applications can manage the device through a secure connection. (The look and feel of a SSH client session is identical to a Telnet session.) You can run either SSH version 1 (SSHv1) or SSH version 2 (SSHv2) on the security device; the commands available depend on the SSH version that you activate.

Syntax

clear

clear ssh
{
all |
enables |
host-key |
pka-key |
sessions
}

exec (SSHv1)

exec ssh tftp pka-rsa [user-name *name_str*] file-name *filename* ip-addr *ip_addr*
[from *interface*]

exec (SSHv2)

exec ssh tftp pka-dsa [user-name *name_str*] file-name *filename* ip-addr *ip_addr*
[from *interface*]

get (SSHv1)

get ssh
[
host-key |
pka-rsa [all | [username *name_str*] [index *number*]] |
report
]

get (SSHv2)

get ssh
[
host-key |
pka-dsa [all | [user-name *name_str*] [index *number*]] |
report
]

set (SSHv1)

```
set ssh
   {
   enable |
   key-gen-time number |
   pka-rsa [ username name_str ] key number1 number2 number3
   }
```

set (SSHv2)

```
set ssh
   {
   enable |
   pka-dsa
      {
      user-name name_str { key string | pka-key-id string } |
      key string
      } |
   pub-key string |
   version { v1 | v2 }
   }
```

Keywords and Variables

all

clear ssh all

all	Clear all SSH sessions, enables, PKA keys, and host keys on device.

enable

set ssh enable
unset ssh enable

enable	Enables the Secure Shell (SSH) task. When issued from a vsys, enables SSH for the vsys.

host-key

get ssh host-key
unset ssh host-key

host-key	The **get** command shows the SSH host key (RSA public key for SSHv1 and DSA public key for SSHv2) for the root or current vsys, including the fingerprint of the host key. The **clear** command deletes the SSH host key for the root or current vsys; SSH must be disabled first before you can delete the host key.

key-gen-time

set ssh key-gen-time *number*
unset ssh key-gen-time

key-gen-time	Specifies the SSHv1 server key regenerating time (in minutes).

pka-dsa

get ssh pka-dsa [...]
set ssh pka-dsa [...]
unset ssh pka-dsa { ... }

pka-dsa — Public Key Authentication (PKA) using Digital Signature Algorithm (DSA) for SSHv2.

- **all** Shows all PKA public keys bound to all users. You must be the root user to execute this option; read-write users and read-only users cannot execute this command.
- **index** *number* allows the admin user and read-only user to view the details of a key bound to the active admin. It also allows the root user to view the details of a key bound to the specified user.
- **key** *string* Binds a PKA key to the current user. Read-only users cannot execute this option.
- **pka-key-id** *string* Binds a PKA key identified by the key ID to the current user. Read-only users cannot execute this option.
- **user-name** *name_str* Specifies the name of the user to bind the PKA key. **file-name** *filename* Specifies the file containing the key to bind to the user. For the get command, **user-name** displays all PKA public keys bound to a specified user *name_str*. Admin users and read-only users can execute this option only if *name_str* identifies the current admin user or read-only user.

Example: The following command binds a hypothetical key to a user named *chris*:

set ssh pka-dsa user-name chris key
AAAAB3NzaC1kc3MAAABBAPrdVkvpSiLMT7NfZJm24pqMU2
FFpO49+LFmbOipIjEYeIWTA4J5...

The following command:

- Loads a key contained in a file named *key_file*
- Takes the file from a server at IP address *172.16.10.11*
- Binds the key to a user named *chris*

exec ssh tftp pka-dsa user-name chris file-name key_file ip-addr 172.16.10.11

pka-key

clear ssh pka-key

pka-key	Deletes all SSH PKA keys on the device.

pka-rsa

get ssh pka-rsa [...]
set ssh pka-rsa [...]
unset ssh pka-rsa { ... }

pka-rsa	Public Key Authentication (PKA) using RSA for SSHv1. ■ **all** Shows all PKA public keys bound to all users. You must be the root user to execute this option; admin users and read-only users cannot execute this command. ■ **index** *number* allows the admin user and read-only user to view the details of a key bound to the active admin. It also allows the root user to view the details of a key bound to the specified user. ■ **key** *number1 number2 number3* Binds a PKA key to the current user. The *number1*, *number2*, and *number3* values represent the key length, the exponent, and the modulus, respectively. Read-only users cannot execute this option. ■ **username** *name_str* Specifies the name of the user to bind the PKA key. **file-name** *filename* Specifies the file containing the key to bind to the user. For the get command, **username** displays all PKA public keys bound to a specified user *name_str*. Admin users and read-only users can execute this option only if *name_str* identifies the current admin user or read-only user.

Example: The following command binds a hypothetical key to a user named *chris*:

**set ssh pka-rsa username chris key 512
6553768752724884489580719560540933919350
0332137246155827968137574227156439706261287933655999926582289
80111611537652715077837089019119296718115311887359071551679**

The following command loads a key:

- Key contained in a file named *key_file*
- File taken from a server at IP address *172.16.10.11*
- Key bound to a user named *chris*

exec ssh tftp pka-rsa username chris file-name key_file ip-addr 172.16.10.11

pub-key

set ssh pub-key *string*
unset ssh pub-key *string*

pub-key	Sets the public key for SSHv2.

report

get ssh report

report	Displays SSHv1 (or SSHv2) key, session, and vsys information for the device on which SSH is currently enabled.

sessions

clear ssh sessions

sessions	Logs out all administrators that currently have active SSH sessions.

version

set ssh version v1 | v2

version	(Available only at the root level.) Sets the version of SSH on the security device. Specify either SSH version 1 or version 2. Before you can set an SSH version, make sure that all keys created with the previous version are removed by executing the **delete ssh device all** command. To clear SSHv2 keys; issue the **clear scs all** command to clear SSHv1 keys.

Defaults

This feature is *disabled* by default. The default key generation time for SSHv1 is *60* minutes.

ssid

Use the **ssid** commands to configure the wireless service set identifier (SSID). You must create an SSID instance before you can configure its parameters.

Syntax

get

get ssid [*name_str*]

set (SSID Instance)

set ssid name *name_str*

set (SSID Authentication)

set ssid *name_str* authentication
{
802.1x auth-server *name_str* |
auto |
open encryption
{
none |
wep
[
key-source { local | server auth-server *name_str* | both auth-server *name_str* }
]
} |
shared-key |
wpa
[rekey-interval { disable | *number* }]
encryption { aes | auto | tkip } auth-server *name_str* |
wpa-auto
[rekey-interval { disable | *number* }]
encryption { aes | auto | tkip } auth-server *name_str* |
wpa-auto-psk
{
passphrase *string* |
psk *key_str* |
}
[rekey-interval { disable | *number* }] encryption { aes | auto | tkip } |
wpa-psk
{
passphrase *string* |
psk *key_str* |
}
[rekey-interval { disable | *number* }]
encryption { aes | auto | tkip } |

```
wpa2
   [ rekey-interval { disable | number } ]
   encryption { aes | auto | tkip } auth-server name_str |
wpa2-psk
   {
   passphrase string |
   psk key_str |
   }
      [ rekey-interval { disable | number } encryption { aes | auto | tkip } ]
}
```

set (SSID Client Isolation)

set ssid *name_str* client-isolation

set (SSID Interface)

set ssid *name_str* interface { *wireless_interface* }

set (SSID WEP Key Configuration)

```
set ssid name_str key-id
      { 1 | 2 | 3 | 4
      length { 104 | 40 }
      [ method { asciitext string | heaxadecimal string [ default ] }
      }
```

set (SSID Broadcast)

set ssid *name_str* ssid-suppression

Keywords and Variables

Variable Parameter

get ssid *name_str*
set ssid name *name_str*
unset ssid name *name_str*

name	Assigns a name to the SSID. The *name_str* can be a maximum of 32 characters. If the name includes a space, the name must be enclosed by quotation marks.

authentication

set ssid *name_str* authentication {...}

authentication — Allows you to set authentication and encryption options for a specific SSID.

- **802.1x auth-server** Specifies the name of the RADIUS server from which the encryption key is retrieved.
- **auto** Specifies that the security device accepts open encryption with Wired Equivalent Privacy (WEP) or shared-key authentication.

- **open encryption** Specifies whether no encryption is performed or WEP encryption is used. In either case, no authentication is performed. You can specify the following options:
  - **none** Specifies that no encryption is performed.
  - **wep** Specifies that WEP encryption is to be used. key-source allows you to select where the WEP key is to be read from; local (from the security device), server (RADIUS server), or both. If you do not specify a key-source, local is selected by default. If the key-source is local or both, you must select a default key. If the key-source is server, the key does not need to exist on the security device.
- **shared-key** Enables shared-key for both authentication and encryption. When this option is specified, the encryption method can only be WEP and you must select a default key.
- **wpa** Enables Wi-Fi Protected Access (WPA) authentication when a RADIUS server is used and sets an optional rekey-interval. If you enable WPA authentication, you also need to configure the RADIUS server.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **auto** Specifies either AES or TKIP encryption.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
  - **auth-server** *name_str* Specifies the RADIUS server that stores authentication information.
- **wpa-auto** Allows WPA or WPA2 as the authentication type.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
    - **auto** Specifies either AES or TKIP encryption.
  - **auth-server** *name_str* Specifies the RADIUS server that stores authentication information.

- **wpa-auto-psk** Allows you to configure the WPA or WPA2 pre-shared key.
  - **passphrase** Sets a passphrase to access the SSID. The string should contain 8 to 63 ASCII characters.
  - **psk** Sets a pre-shared key to access the SSID. The key must be a 256-bit (64 characters) hexadecimal value.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
    - **auto** Specifies either AES or TKIP encryption.
- **wpa-psk** Allows you to configure the WPA pre-shared key on the security device.
  - **passphrase** Sets a passphrase to access the SSID. The string should contain 8 to 63 ASCII characters.
  - **psk** Sets a pre-shared key to access the SSID. The key must be a 256-bit (64 characters) hexadecimal value.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
    - **auto** Specifies either AES or TKIP encryption.
- **wpa2** Enables Wi-Fi Protected Access 2 (WPA2) authentication when a RADIUS server is used and sets an optional rekey-interval. If you enable WPA authentication, you also need to configure the RADIUS server.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
    - **auto** Specifies either AES or TKIP encryption.
  - **auth-server** *name_str* Specifies the RADIUS server that stores authentication information.

- **wpa2-psk** Allows you to configure the WPA2 pre-shared key on the security device.
  - **passphrase** Sets a passphrase to access the SSID. The string should contain 8 to 63 ASCII characters.
  - **psk** Sets a pre-shared key to access the SSID. The key must be a 256-bit (64 characters) hexadecimal value.
  - **rekey-interval** Sets the group key update interval, which can range from 30 to 4,294,967,295 seconds. The default value is 1800 seconds. You can also specify **disable** if you are not using key updates.
  - **encryption** Specifies the encryption used between the security device and wireless clients in the subnetwork. You can specify the following options:
    - **aes** Specifies Advanced Encryption Standard (AES), used by WPA2 devices.
    - **tkip** Specifies Temporal Key Integrity Protocol (TKIP), used by WPA devices.
    - **auto** Specifies either AES or TKIP encryption.

Example: The following examples set different types of authentication and encryption methods for the SSID named example1.

set ssid example1 authentication auto
set ssid example1 authentication open encryption wep

client-isolation

set ssid *name_str* client-isolation
unset ssid *name_str* client-isolation

client-isolation	Prevents wireless clients on the same subnetwork of the SSID from accessing each other. Note that intra-zone blocking, which you can configure with the **set zone** command, blocks traffic between an SSID and a wired or wireless subnetwork.

interface

set ssid *name_str* interface { *wireless_interface* }
unset ssid *name_str* interface

interface	Binds a wireless interface to an SSID and activates the SSID. The number of wireless interfaces you can bind and activate depends on the security device.

key-id

set ssid *name_str* key-id { 1 | 2 | 3 | 4 } ...
unset ssid *name_str* key-id { 1 | 2 | 3 | 4 }

key-id	Enables WEP key configuration and sets the WEP key value. The value range is 1 through 4.
length	Specifies the length of the encryption key (in bits): ■ **40-bit** Enter 10 hexadecimal digits or 5 ASCII characters. ■ **104-bit** Enter 26 hexadecimal digits or 13 ASCII characters.
method	Sets the string type: **asciitext** *string* or **hexadecimal** *string*. The default method is hexadecimal. Use the default keyword to specify the default key. If you do not specify a default key, the key that is entered first is the default.

Example: This examples sets the SSID example with a key-id of 1, key length of 40 bits, and ASCII password abcde.

set ssid example key-id 1 length 40 method asciitext abcde

ssid-suppression

set ssid *name_str* ssid-suppression
unset ssid *name_str* ssid-suppression

ssid-suppression	Disables broadcasting of SSIDs in beacons that are advertised by the security device. If SSID broadcasting is disabled, only wireless clients that know of the SSID are able to associate. By default, SSIDs are broadcast in beacons.

ssl

Use the **ssl** commands to configure a Secure Sockets Layer (SSL) connection, or to display the SSL configuration on a security device.

Secure Sockets Layer (SSL) is a set of protocols that can provide a secure connection between a Web client and a webserver communicating over a TCP/IP network.

Syntax

get

get ssl [ca-list | cert-list]

set

set ssl
{
cert *number* |
enable |
encrypt { { 3des | des } sha-1 | { rc4 | rc4-40 } md5 }
port *port_num*
}

Keywords and Variables

ca-list | cert-list

get ssl ca-list
get ssl cert-list

ca-list | cert-list	Displays currently configured Certificate Authorities (**ca-list**) or currently available certificates (**cert-list**).

Example: The following command displays the SSL certificate list:

get ssl cert-list

cert

set ssl cert *number*
unset ssl cert

cert	Specifies that the named certificate is required.

enable

set ssl enable
set ssl enable
unset ssl enable

enable	Turns on SSL.

encrypt

set ssl encrypt
{ 3des | des } sha-1 |
{ rc4 | rc4-40 } md5
unset ssl encrypt

encrypt — Enables encryption over the SSL connection.

- **3des** Sets the 3DES security level.
- **des** Sets the DES security level.
- **rc4 md5** Sets the RC4 MD3 security level.
- **rc4-40 md5** Sets the RC4-40 MD3 security level.

Example: The following command specifies triple-DES encryption with SHA-1 authentication hashing:

set ssl encrypt 3des sha-1

port

set ssl port *port_num*
unset ssl port

port	Specifies the SSL port number.

Example: The following command changes the SSL port to 11533:

set ssl port 11533

Defaults

The default SSL port is *443*.

switch

Use the **switch** commands to test the switch module on some devices.

Syntax

exec switch
{
reset-counter |
reset-statistic |
snoop { rx *number* | tx *number* }
}

Keywords and Variables

switch

exec switch { ... }

switch	Executes switch module testing.

reset-counter

exec switch reset-counter

reset-counter	Resets the rx and tx counters.

reset-statistic

exec switch reset-statistic

reset-statistic	Clears all statistics.

snoop

exec switch snoop { rx *number* | tx *number* }

snoop	Sets the memory rx and tx dump size.

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* | *name_str*]

config	Defines the configuration settings for the syslog utility. The **{ name_str** | **ip_addr }** parameters define the hose name or the IP address of the syslog host device. You can define up to four syslog hosts. Specifying an IP 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 { ... { ... } }

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: The following command sets the syslog host configuration to report all logs:

set syslog config 172.16.20.249 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 all log entries to the syslog host.

port

set syslog config { *name_str* | *ip_addr* } port *port_num*
unset syslog config { *name_str* | *ip_addr* } port

port	Defines the port number (*port_num*) 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

set syslog config { *ip_addr* | *name_str* } transport tcp
unset syslog config { *ip_addr* | *name_str* } transport

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 the following:

- Descriptive indices of the ScreenOS operating system, including serial number, control number, software number, and image-source filename
- Descriptive indices of the 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 security device. Most of this information consists of the current authentication and routing settings.

Syntax

get

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

Variable Parameters

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

Variable Parameters

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.) For example, **unset timer 1**.

action

set timer *date time* action reset
unset timer *id_num*

action reset	Automatically resets the device at the configured time.

Example: The following command configures the security device to reset at a given time and date:

set timer 1/31/2007 19:00 action reset

trace-route

Use the **trace-route** commands to display the route to a host.

Syntax

trace-route { *ip_addr* | *name_str* } [hop *number* [time-out *number*]]

Keywords

Variable Parameters

trace-route *ip_addr*
trace-route *name_str*

ip_addr | *name_str*	The IP address (*ip_addr*) or object name (*name_str*) of the host.

hop

trace-route { *ip_addr* | *name_str* } hop *number* [...]
trace-route { *ip_addr* | *name_str* } hop *number* time-out *number*

hop	The maximum number of trace route hops (*number*) to evaluate and display. ■ **time-out** Specifies the amount of time in seconds (*number*) to elapse before abandoning the route trace.

Example: The following command performs a trace-route operation:

- Evaluates and displays up to four route trace hops
- Sends the output to a host with IP address 172.16.10.10
- Specifies a timeout value of four seconds

trace-route 172.16.10.10 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 on 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 device.

Syntax

get

```
get traffic-shaping
    [
    dscp-class-selector |
    interface [ interface ] |
    ip_precedence |
    mode |
    statistics
    ]
```

set

```
set traffic-shaping
    {
    dscp-class-selector |
    ip_precedence
      { number1 number2 number3 number4 number5 number6 number7 number8 |
    mode { auto | off | on }
    }
```

Keywords and Variables

dscp-class-selector

get traffic-shaping dscp-class-selector
set traffic-shaping dscp-class-selector
unset traffic-shaping dscp-class-selector

dscp-class-selector	Subsumes IP precedence into class selector codepoints, ensuring that priority levels set with **ip_precedence** are preserved and handled correctly by downstream routers.

interface

get traffic-shaping interface [*interface*]

interface	Displays the traffic shaping information 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 traffic shaping mode function for the system. The default mode is **Auto**.

- **auto** Specifies that traffic shaping be enabled automatically only when there is a policy that has either ingress policing or traffic shaping enabled.
- **off** Specifies that shaping is not enabled even if there is a policy that has either ingress policing or traffic shaping enabled.
- **on** Specifies that shaping is enabled regardless of the presence of a policy that has ingress policing or shaping enabled.

statistics

statistics	Displays statistical information about traffic shaping and traffic policing.

url

Use the **url** commands to enable or disable web filtering for use in policies and to configure and display web-filtering settings.

ScreenOS supports two types of web filtering:

- Integrated

  Some security devices support an integrated web-filtering solution that employs Content Portal Authority (CPA) servers from SurfControl.

NOTE: Integrated web filtering requires you to install a license key on your security device.

- Redirect

  Some security devices support a web-filtering solution that employs SurfControl or Websense services to a SurfControl or Websense server.

To run either of the web-filtering features on the security device, perform the following steps:

1. Select the protocol.

   For example, the **set url protocol type { sc-cpa | scfp | websense }** command selects the protocol.

2. Initiate the web-filtering context.

   Executing the **set url protocol { sc-cpa | scfp | websense }** command places the CLI in the web-filtering routing context and redirects web filtering to the SurfControl or Websense servers. Once you initiate the web-filtering context, all subsequent command executions apply to the web-filtering feature.

For more information and examples, refer to the *Concepts & Examples ScreenOS Reference Guide.*

Syntax

get

get url [all | vsys-name *vsys_name*]

set (root and vsys level)

set url protocol
{
type { sc-cpa | scfp | websense }
sc-cpa |
scfp |
websense
}

set (within the protocol context)

set {
account *name_string* |
config { disable | enable } |
fail-mode { block | permit } |
deny-message { *string* | use-server }
server {
{ *ip_addr* | *dom_name* } |
host *string* |
port_num |
timeout_num |
src-interface *interface* |
} |
use-root |
use-vsys
}

Integrated Web-Filtering (SC-CPA) Commands

To run the integrated web-filtering feature (SurfControl Content Portal Authority-SC-CPA) on the security device, you must select the protocol and initiate the web-filtering context as follows:

```
set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa)->
```

The following **set** commands are executable in this web-filtering context (url:sc-cpa):

cache	Use the **set cache** command to enable caching. You can also change the cache size or timeout value.
cate-list-query-interval	Use the **set cate-list-query-interval** command to specify the interval at which the device queries the SurfControl CPA server for categorization updates.
category	Use the **set category** command to create a category or to add a URL to a category. You can add up to 20 URLs to a category.
enable	Use the **set enable** command to enable web filtering using the SurfControl Content Portal Authority (CPA) servers.
fail-mode	Use the **set fail-mode** command to block or permit all requests when the web-filtering server fails.
profile	Use the **set profile** command to create a new web-filtering profile or to add a category to a profile.
server (integrated web-filtering)	Use the **set server** command to define the primary web-filtering server.

NOTE: The **enable**, **fail-mode**, and **profile** commands can be set in vsys mode. The rest of the commands in integrated web filtering are read-only.

The following **get** commands are executable in the web-filtering context (url:sc-cpa):

category	Use the **get category** command to display the URL categories.
ns-profile	Use the **get ns-profile** command to display the default web-filtering profile.
profile	Use the **get profile** command to display all web-filtering profiles.
server (integrated web-filtering)	Use the **get server** command to display information from the primary web-filtering server.

Redirect Web-Filtering (SCFP and Websense) Commands

To run the redirect web-filtering feature on the security device, you must select the protocol and initiate the web-filtering context as follows:

Redirecting to SurfControl Servers	Redirecting to Websense Servers
set url protocol type scfp set url protocol scfp (url:scfp)->	set url protocol type websense set url protocol websense (url:websense)->

Security devices with virtual systems support up to eight different URL-filtering servers—one server reserved for the root system, which can be shared with an unrestricted number of virtual systems; and seven URL-filtering servers for private use by the virtual systems. A root-level administrator can configure the URL-filtering module at the root and virtual system (vsys) levels. A vsys-level administrator can configure the URL module for his or her own vsys if that vsys has its own dedicated URL-filtering server. If the vsys-level administrator uses the root URL-filtering server settings, that admin can see—but not modify—the root-level URL-filtering settings.

The following **set** commands are executable in the redirect web-filtering context **(root and vsys)**:

account	Use the **set account** command to set the web-filtering account.
config	Use the **set config** command to enable or disable web filtering at the device level for use in policies. By itself, enabling web filtering at the device level does not activate it. You must enable web filtering at both the device and the policy levels in order to apply filtering to URL requests.
deny-message	Use the **set deny-message** command to customize the blocked URL message. Specify the message source that the device delivers to the clients when URLs are blocked.
fail-mode	Use the **set fail-mode** command to block or permit all requests when the web-filtering server fails.
server (redirect web-filtering)	Use the **set server** command to define the primary web-filtering server. Use the **set url src-interface** command to define to which server the devices sends the URLs to be categorized.
use-root	Use the **set use-root** command to instruct a vsys to share a web-filtering server that was defined at the root level.
use-vsys	Use the **set use-vsys** command to instruct the vsys to use the web-filtering server that was defined for that vsys.

Keywords and Variables

account

set account *name_str*

name-str	Sets a name for the web-filtering server account. You must be in the vsys level to execute this command.

Example: Set up a web-filtering server account for the marketing department.

```
set url protocol type scfp
set url protocol scfp
(url:scfp) -> set account mtg-server
```

cache

set cache { enable | size *number* | timeout *number* }
unset cache { enable | size | timeout }

enable	Enables the device to cache the categorization of URLs.
size	Specifies the memory size of the categorization cache.
timeout	Specifies the number of hours the device stores entries in the categorization cache.

Example: Set up the device to cache the URL categorization in a 10 MB cache size, and store the URLs in the cache for 24 hours.

```
set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa) -> set cache enable
(url:sc-cpa) -> set cache size 20
(url:sc-cpa) -> set cache timeout 24
(url:sc-cpa) -> exit
```

cate-list-query-interval

set cate-list-query-interval *number*
unset cate-list-query-interval

cate-list-query-interval	Specifies the interval at which the device queries the SurfControl CPA server for categorization updates.

Example: Set up the device to query the Websense server every 60 minutes for categorization updates.

```
set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa) -> set cate-list-query-interval 60
(url:sc-cpa) -> exit
```

category

set category *name* url *url_str*
get category [*name* | pre | user]
unset category *name* [url *url_str*]

category	Specifies the category you are creating or to which you are adding a URL.
pre	Displays the predefined categories.
url	Specifies the URL you are adding.
user	Displays the user-defined categories.

Example: Configure a customized URL category and add URLs to it.

```
set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa) -> set category name banks url mybank.com
(url:sc-cpa) -> set category name banks url yourbank.com
(url:sc-cpa) -> exit
```

config

set config { disable | enable }
unset config

config { disable | enable }	Disables or enables web filtering at the device level for use in policies. By itself, enabling web filtering at the device level does not activate it. You must enable web filtering at both the device and the policy levels in order to apply filtering to URL requests.

Example: Enable web filtering at the policy level.

```
set url protocol type scfp
set url protocol scfp
(url:scfp) -> set config enable
```

enable

set enable
unset enable

enable	Enables web filtering using the SurfControl CPA servers.

Example: Enable integrated web filtering.

```
set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa) -> set enable
```

fail-mode

set fail-mode { block | permit }
unset fail-mode

fail-mode { block | permit }	If the connection between the device and the Websense server is lost, the device either blocks or permits all HTTP requests to which a policy requiring web filtering applies. The default fail-mode behavior is to block HTTP requests.

Example: Enable redirect web filtering to block HTTP requests when the connection to the Websense server goes down.

set url protocol type websense
set url protocol websense
(url:websense) -> set failmode block
(url:websense) -> exit

deny-message

set deny-message { *string* | use-svr }
unset deny-message

message	Specifies the source of the message that the device delivers to clients when URLs are blocked—the device or the Websense server. ■ *string* Defines a custom message from the device, 1 to 500 characters in length to be sent to the client that is blocked from reaching a URL. ■ **use-svr** Defines a message from the server to be sent to the client that is blocked from reaching a URL.

Example: The following command defines the URL blocking message "This site is blocked."

set url protocol type scfp
set url protocol scfp
(url:scfp) -> set deny-message "This site is blocked."

ns-profile

get ns-profile

ns-profile	Displays the predefined profile. You must initiate the web-filtering context before you can execute this command.

profile

set profile *string1* { other block | permit } | *string2* { block | permit | black-list | white-list }
unset profile *string1* [other | *string2* | black-list | white-list]
get profile [*string*]

profile *string1*	Specifies the profile you are creating or updating. The default *string1* is **ns-profile**. You must initiate the web-filtering context before you can execute this command.
other	Specifies the Other category. Use this keyword to define the action for this category.
block	The device blocks access to URLs in the specified category.
permit	The device permits access to URLs in the specified category.
string2	Specifies the category for which you are defining an action.
black-list	The device blocks access to URLs in this category.
white-list	The device permits access to URLs in this category.

protocol

set url protocol type { sc-cpa | scfp | websense }
set url protocol { sc-cpa | scfp | websense }
unset url protocol type

type sc-cpa | scfp | websense	Indicates which web-filtering protocol you are configuring: ■ **sc-cpa** Integrated web filtering with the SurfControl servers. ■ **scfp** Redirect web filtering with the SurfControl servers. ■ **websense** Redirect web filtering with the Websense servers. For more information about web-filtering protocols, refer to the *Concepts & Examples ScreenOS Reference Guide*.
protocol sc-cpa | scfp | websense	Initiates the following web-filtering context: **(url: sc-cpa)** - > **(url: scfp)** - > **(url: websense)** - >

server (redirect web-filtering)

set server { { *ip_addr* | *dom_name* } *port_num number* | src-interface *interface* }
set src-interface *interface*
unset server
unset src-interface

server { *ip_addr* | *dom_name* }	Defines the following connection parameters for the web-filtering server: ■ *ip_addr* | *dom_name* Sets the IP address or DNS name of the web-filtering server. ■ *port_num* Sets the port number on which the device communicates with the web-filtering server. The default port number is 15868. ■ *number* Sets the timeout interval, in seconds, that the device waits for a response from the Websense filter. If Websense does not respond within the time interval, the device either blocks the request or allows it, as you choose. The default is 10 seconds.
src-interface *interface*	Specifies the source interface that the 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.

Example: The following command sets the IP address, port number, and timeout value for the web-filtering server (the port number and timeout interval use the default values):

set url protocol type scfp
set url protocol scfp
(url:scfp) -> set url server 1.2.2.20 15868 10
(url:scfp) -> exit

server (integrated web-filtering)

set server { america | asia | europe }
unset server { america | asia | europe }
get server

server	Defines the primary CPA server to which the device sends URLs for categorization. You must initiate the web-filtering context before you can execute this command.

Example: The following commands define the asia server to be the primary CPA server for web filtering:

set url protocol type sc-cpa
set url protocol sc-cpa
(url:sc-cpa) -> set server asia

use-root

set url use-root

use-root	When this command is entered in a virtual system (vsys), it instructs the vsys to share the web-filtering server defined at the root level.

Example: Configure a vsys to use the web-filtering settings of the root-vsys.

```
device-> set vsys v1
device(v1)-> set url protocol type websense
device(v1)-> set url protocol websense
device(v1/url:websense) -> set use-root
device(v1/url:websense) -> exit
```

use-vsys

set url use-vsys

use-vsys	When this command is entered in a virtual system (vsys), it instructs the vsys to use the web-filtering server defined for that vsys.

usb-device

Use the **usb-device** commands to execute a USB storage device inserted in the USB host module found on some devices.

Syntax

exec usb-device [stop]

Keywords and Variables

usb-device

exec usb-device
exec usb-device stop

usb-device	Executes or stops the use of a USB storage device.

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 [ share-limit number ] |
      ip string [ share-limit number ] |
      u-fqdn name_str [ share-limit number ]
      } |
    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 ] [ wan ] [ xauth ] } |
    uid id_num
    }
```

Keywords and Variables

Variable Parameters

get user *name_str*
set user *name_str* { ... }
unset user *name_str* [...]

user	Defines the user's name (*name_str*).

all

get user all

all	Displays the following information for all the entries in the internal user database: ■ User ID number ■ User name ■ Status (enabled or disabled) ■ User type ■ IKE ID types—email address, IP address, or domain name—and IKE identity ■ Groups to which a user belongs

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 on the user, identified by id_num. This option displays the same information as **get user** *name_str* option.

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 { *string* | *name_str* } Adds and defines an AutoKey IKE dialup user.

- **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 nonempty 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.
- **fqdn** *name_str* The Fully Qualified Domain Name, the complete string, such as www.juniper.net.
- **ip** *string* The IP address of the dialup user, such as 192.168.1.1.
- **u-fqdn** *name_str* Specifies the dialup user identity, usually equivalent to an email address such as admin@acme.com.

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 on 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 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* |
ipaddr *ip_addr* |
ippool *name_str* |
wins1 *ip_addr* |
wins2 *ip_addr*
}
unset user *name_str* remote-settings { dns1 | dns2 | ipaddr | ippool | wins1 | wins2 }

remote-settings	Sets the remote settings for the user. ■ **dns1** | **dns2** Specifies the IP address (*ip_addr*) of the primary and secondary DNS servers. ■ **ipaddr** Specifies the static IP address (*ip_addr*) for the user. ■ **ippool** Specifies the named L2TP IP pool (*name_str*), 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 (*ip_addr*) 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] [wan] [xauth] }
unset user name_str type {...}

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**. Type **wan** is used for PPP and MLPPP encapsulated data links only. The **type wan** command, defines the user as a WAN user. If CHAP or PAP authentication is configured for the PPP data link, the username and password for the peer device must be configured as a WAN user type.

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 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] [l2tp] [xauth] } |
user *name_str*
}

Keywords and Variables

Variable Parameter

get user-group *name_str*
set user-group *name_str* { ... }
unset user-group *name_str* [...]

name_str	Specifies the name of the user group.

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 {...}
unset 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. ■ **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	Adds or removes the named user (*name_str*) to the specified user group.

Example: The following example does the following:

- Creates a new authentication user named *guest*
- Authenticates user group named *Corp_Dial* with ID *1010*
- Adds a user to the user group

set user guest password JnPc3g12
set user-group Corp_Dial location local
set 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 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

Variable Parameters

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.

CAUTION: After you execute this command, you must restart the device. This command changes the functionality of the VIP. Switching back and forth between enabling and disabling the **multi-port** modes is not recommended.

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.

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 Internet Key Exchange (IKE) 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

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*
}

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_addr1*
  [outgoing-interface *interface* [local-address *ip_addr2*]]
    {
    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* }
        ]
    }

Keywords and Variables

Variable Parameters

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-1** Specifies 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. **Note:** When you omit the **auth** keyword, the device automatically uses the **null** switch. This is not advisable, because it may leave IPSec vulnerable to attack. **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

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* [local-address *ip_addr*] { ... }

outgoing-interface	Defines the interface through which the security device sends traffic for this Manual Key VPN. The **local-address** *ip_addr* value specifies the IP address of the outgoing interface for reverence by external devices. For more information on interfaces, see “Interface Names” on page A-I.

Example: The following command uses a manual tunnel.

- External gateway device IP address 1.1.1.1
- Ethernet1 as the outgoing interface, identified to outside hosts as IP address 2.2.2.2
- Specified encryption algorithm 3DES
- Password “swordfish”

set vpn tun1 manual 20001 20022 gateway 1.1.1.1 outgoing-interface ethernet1 local-address 2.2.2.2 esp 3des password swordfish

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 IP address that sends and receives traffic through the tunnel. ■ **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: The following command creates a VPN proxy configuration for a VPN (*Sales*) with the HTTP service:

set vpn Sales proxy-id local-ip 10.1.1.0/24 remote-ip 10.2.2.0/24 HTTP

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. The standard proposal provides settings recommended by Juniper Networks.

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 a 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 be placed in a VPN group or removed from it.

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 the 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 ScreenOS-specific MIB extension files into the application. The MIB extension files are on the 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

For information about setting protocol-specific parameters, see section of this *CLI Reference Guide*. Protocol-specific commands for RIP, OSPF, IGMP, and PIM are listed alphabetically by name in this *CLI Reference Guide*.

Syntax

clear

clear vrouter *vrouter*
{
mroute { all | mgroup *ip_addr* [source *ip_addr*] [iif *interface*] |
protocol bgp neighbor *ip_addr* { soft-in | soft-out }
statistics
}

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 |
    mcore [ cachemiss ] |
    mroute [ brief ]
      [
      mgroup ip_addr brief |
      source ip_addr
        [
        brief |
        iif interface
        ]
      ]
    preference |
    protocol { bgp | ospf | rip | pim} |
```

NOTE: For more information about the **protocol { bgp | ospf | rip }** options, see the **bgp**, **ospf**, and **rip** command descriptions.

```
    route
      [
      id id_num |
      ip ip_addr |
      prefix ip_addr/mask |
      protocol { bgp | connected |discovered | imported | ospf | rip | static }
      source
        [ ip_addr
          [ interface interface [ gateway ip_addr ] | vrouter vrouter ]
        ]
        [
        id id_num |
        in-interface [ interface ] |
        ip ip_addr |
        prefix ip_addr/mask |
        ] |
      summary
      ] |
    route-lookup preference |
    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*
{ permit | deny } { ip ip_addr/mask | default-route } *number* |
add-default-route vrouter untrust-vr |
adv-inact-interface
auto-route-export |
default-vrouter |
export-to | import-from
vrouter *name_str* route-map *name_str* protocol
{ bgp | connected | discovered | imported | ospf | rip | static }
ignore-subnet-conflict |

NOTE: For more information on the **protocol { bgp | ospf | rip }** options, see the **bgp**, **ospf**, and **rip** command descriptions.

max-ecmp-routes *number* |
max-routes *number* |
mroute
{
max-entries *number* |
mgroup *ip_addr* source *ip_addr* iif *interface* oif *interface* out-group *ip_addr*|
multiple-iif-enable |
negative-cache [timer *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 } |

NOTE: For more information on the **protocol { bgp | ospf | pim | rip }** options, see the **bgp**, **ospf**, **pim**, and **rip** command descriptions.

route [source] [in-interface *interface*] *ip_addr/mask*
{
interface *interface*
[gateway *ip_addr*] [metric *number*] [permanent]
[preference *number*][tag *id_num*] |
vrouter *name_str*
} |
route-lookup
preference
[
destination-routing *number* |

```
   sibr-routing number |
   source-routing number |
   ] |
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 |
      origin { igp | incomplete }
      preserve preference |
      preserve metric |
      tag { number | ip_addr } |
      weight number
      ]
router-id { id_num | ip_addr } |
sharable |
sibr-routing enable |
snmp trap private |
source-routing enable
]
```

Keywords and Variables

Variable Parameter

clear vrouter *vrouter* protocol bgp neighbor *ip_addr* soft-out
set vrouter *name_str*

ip_addr	Specifies an IPv4 address of BGP neighbor.
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 BGP routing context, and execute the context-dependent command **get config**.

set vrouter trust-vr
device(trust-vr)-> **set protocol bgp**
device(trust-vr/bgp)-> **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* }
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 *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.

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 on Route mode and NAT mode, refer to the *Concepts & 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.

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.
- **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-ecmp-routes

set vrouter *name_str* max-ecmp-routes *number*
unset vrouter *name_str* max-ecmp-routes

max-ecmp-routes	Specifies the maximum number of equal cost multipath (ECMP) routes to the same destination network. Enter a value between 1 and 4 (1 is the default).

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.

mcore

get vrouter *name_str* mcore [cachemiss]

mcore	Displays multicast routing information for each interface on which a multicast routing protocol is enabled.
cachemiss	Displays the current multicast cachemiss data.

mroute

get vrouter *name_str* brief
get vrouter *name_str* mroute mgroup *ip_addr1* brief
get {...} mroute mgroup *ip_addr1* source *ip_addr2* [brief | iif *interface1*]
set vrouter *name_str* mroute max-entries *number*
set {...} mroute mgroup *ip_addr1* source *ip_addr2* iif *interface1* oif *interface2*
set {...} mroute mgroup *ip_addr1* {...} out-group *ip_addr3*
set vrouter *name_str* mroute multiple-iif-enable
set vrouter *name_str* negative-cache [timer *number*]
unset vrouter *name_str* mroute max-entries
unset {...} mroute mgroup *ip_addr1* source *ip_addr2* iif *interface1* oif *interface2*
unset vrouter *name_str* mroute multiple-iif-enable
unset vrouter *name_str* negative-cache [timer *number*]

brief — Displays summary information.

max-entries — Specifies the maximum number of multicast routes allowed in the multicast routing table.

mroute — Configures a static multicast route in the specified virtual router.

- *ip_addr1* is the multicast group address of the route
- *ip_addr2* is the source address of the multicast data
- *interface1* is the incoming interface of the multicast data
- *interface2* is the outgoing interface of the multicast data
- *ip_addr3* is the multicast group address on the outgoing interface

multiple-iif-enable — Permits multiple multicast routes for the same source and group.

negative-cache — Creates negative multicast routes if the protocol that owns the interface on which the packet was received cannot create a forwarding multicast route. The security device drops packets when they need to go on a negative multicast route. You can also set the timer value to specify the duration, in seconds, that the security device maintains the entries in the negative cache. The security device removes the entry in the negative cache when it receives information enabling it to create a forwarding multicast route entry.

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.
- **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 | pim }
set vrouter *name_str* protocol { bgp | ospf | rip | pim }
unset vrouter *name_str* protocol { bgp | ospf | rip | pim }

protocol

Places the security device in the context of the specified protocol : BGP, OSPF, RIP or PIM. (For information on these contexts, see the **bgp**, **ospf**, **rip** or **pim** 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] [in-interface *interface*] *ip_addr/mask* [...]
unset vrouter *name_str* route [source] [in-interface *interface*] *ip_addr/mask* [...]

route

Configures routes for the routing table for the virtual router.

- *ip_addr/mask* Specifies the IP address that appears in the routing table.
- **gateway** *ip_addr* Specifies the gateway for the next hop.
- **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.
- **in-interface** *interface* For source interface-based routes, specifies the interface on which a packet arrives on the security device. You can then forward that traffic to either a routed interface or to a virtual router.
- **interface** *interface* Specifies the interface on which a packet for this route is to be forwarded.
- **ip** *ip_addr* Displays the route for the specified IP address.
- **metric** *number* Specifies the cost of the route. Specify a value between 1 and 65535.
- **permanent** Specifies that the route is kept active when the interface is down or the IP address is removed from the interface.
- **preference** *number* Specifies the preference value for the route. Specify a value between 0 and 255.
- **prefix** *ip_addr/mask* Displays the routes within the specified subnet address.
- **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*
  - *ip_addr/netmask* **interface** *interface* **gateway** *ip_addr* sets a gateway as the next hop.
  - **vrouter** *vrouter* sets a virtual router as the next hop.
- **summary** Displays a summary of the routes.
- **tag** *number* For destination-based routes, specifies the tag for this route. The tag can be used as a filter when redistributing routes (see the **route-map** keyword). Specify a value between 1 and 65535.
- **vrouter** *name_str* Specifies a virtual router as the next hop.

Example 1: This example sets a source based route. Traffic enters at ethernet1/1, and the next hop is set to be the virtual router *untrust-vr*.

set vrouter trust-vr route source ethernet1/1 10.2.2.1/24 vrouter untrust-vr

Example 2: This example sets a source interface-based route (SIBR). Traffic enters at ethernet1/1, and the next hop is set to be the virtual router *untrust-vr*.

set vrouter trust-vr route source in-interface ethernet1/1 10.2.2.1/24 vrouter untrust-vr

route-lookup preference

get vrouter *name_str* route-lookup preference
set vrouter *name_str* route-lookup preference [destination-routing *number*] [sibr-routing *number*] [source-routing *number*]
unset vrouter *name_str* route-lookup preference

route-lookup preference

Configures the order in which route lookups occur in the virtual router. The route lookup type that has the highest preference value is performed first, followed by the next highest preference value. The route lookup type that has the lowest preference value is performed last. Enter a number between 1-255 for the preference.

- **destination-routing** *number* Specifies the preference for route lookups based on destination IP address. The default value is 1.
- **sibr-routing** *number* Specifies the preference for route lookups based on source interface. The default value is 3.
- **source-routing** *number* Specifies the preference for route lookups based on source IP address. The default is 2.

route-map

get vrouter *name_str* route-map […]
set vrouter *name_str* { … } vrouter *name_str* route-map
{ name *name_str* | *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 **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 value to increment the metric for the matching route. For RIP routes, you can use this option for routes that are advertised or routes that are learned. For other routes, you can use this option to routes that are exported into another virtual router.

- **origin** Specifies the origin of a route advertised by BGP
- **preserve metric** Specifies that the metric value for the matching route is preserved when the route is exported to another virtual router.
- **preserve preference** Specifies that the preference value for the matching route is preserved when the route is exported to another virtual router.
- **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.

sibr-routing enable

set vrouter *name_str* sibr-routing enable
unset vrouter *name_str* sibr-routing enable

source- routing enable	Directs the virtual router to perform routing table lookups based on the source interface.

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.)

soft-in

clear vrouter *trust-vr* protocol bgp neighbor *ip_addr* soft-in

soft-in	Enables a soft reset and generates an inbound update from a BGP neighbor. A soft reset allows the application of a new or changed policy without clearing an active BGP session. The route-refresh feature occurs on a per-neighbor basis and does not require preconfiguration or extra memory.

soft-out

clear vrouter *trust-vr* protocol bgp neighbor *ip_addr* soft-out

soft-out	Enables a soft reset and sends a new set of updates to a BGP neighbor. A soft reset allows the application of a new or changed policy without clearing an active BGP session. The route-refresh feature occurs on a per-neighbor basis; and outbound resets don't require preconfiguration or routing table update storage.

source-routing enable

set vrouter *name_str* source-routing enable
unset vrouter *name_str* source-routing enable

source- routing enable	Directs the virtual router to perform routing table lookups based on source IP address.

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 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 adminstrators 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*]

set

```
set vsys name_str
    [
    vrouter
      [
      name [ name_str ] [ vsd number ] |
      share [ name_str ] [ vsd number ] |
      vsd number
      ] |
    vsd number
    ]
```

Keywords and Variables

Variable Parameters

get vsys [*name_str*]
set vsys *name_str*
unset vsys *name_str*

name_str	Defines the name of a virtual system (vsys) and automatically places the root level admin within the vsys. Subsequent commands configure the newly created vsys.

Example: The following command creates a virtual system named *vsys1* and switches the console to the new virtual system:

set vsys vsys1

vrouter

set vsys *name_str* vrouter [name [*name_str*] [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 *name_str* for the virtual router or the nsrp vsd number. ■ **share** Specifies a shared root-level virtual router to use as a default router for a specified vsys with name *name_str* or nsrp vsd *number*. ■ **vsd** *number* See vsd on page 627.

Example 1: The following command creates a vsys named *Acme_Org*, creates a virtual router named *Acme_Router* with vsd number *3*, and switches the console to the new virtual system:

set vsys Acme_Org vrouter name Acme_Router vsd 3

Example 2: The following command creates a vsys 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. The VSD number can be 1 through 8. 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 on VSD groups, refer to the *Concepts & Examples ScreenOS Reference Guide*.

Example: The following command creates a vsys named *Acme_Org*, creates a virtual router named *Acme_Router*, creates a VSD number *5*, and switches the console to the new virtual system:

set vsys Acme_Org vrouter vsd 5

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 specify authentication in policy definitions (see “auth” on page 55).

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 *wa_serv1*:

set webauth server wa_serv1

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 host name.

port

set webtrends port *port_num*
unset webtrends port

port *port_num*	Specifies the WebTrends host port.

wlan

Use the **wlan** commands to configure the Wireless LAN features.

Syntax

exec

exec wlan { find-channel | reactivate | site-survey }

get

get wlan [acl]

set

```
set wlan
    {
    acl { mac_addr { allow | deny } | mode { enable | strict } } |
    advanced
        { aging-interval { disable | number} |
        beacon-interval { number } |
        burst-threshold { number } |
        cts-mode { auto | off | on } |
        cts-rate { 1 | 11 | 2 | 5.5 } |
        cts-type { cts-only | cts-rts } |
        dtim-period { number } |
        fragment-threshold { number } |
        long-preamble |
        rts-threshold { number } |
        slot-time long } |
    antenna { a | b | diversity } |
    channel { auto | number } |
    country-code { name_str } |
    mode { 11b | 11g [ 11g-only ] } |
    transmit { power { eight | full | half | minimum | quarter } |
            rate { 1 | 11 | 12 | 18 | 2 | 24 | 36 | 48 | 5.5 | 54 | 6 | 9 | auto
                }
            }
    }
```

Keywords and Variables

acl

set wlan acl { *mac_addr* { allow | deny } | mode { disable | enable | strict } }
get wlan acl

acl	Allows or denies access for the specified MAC address (mac_addr). You can specify a maximum of 128 MAC addresses.
mode	Sets the wireless client restriction. ■ disable: The device does not check for restricted clients. ■ enable: Wireless clients that match the deny list are not allowed: all other clients are allowed. ■ strict: Only wireless clients that match the allow list are allowed; all other clients are denied.

Example: this example sets the WLAN to only allow the wireless client with MAC address 000bdfd781f9 to access the security device.

set wlan acl 000bdfd781f9 allow mode strict

advanced

set wlan advanced { ... }
unset wlan advanced { ... }

advanced — Allows you to configure the following advanced ration settings.

- aging-interval Sets the aging vale for wireless clients and bridge entities. Value range is 60 to 1,000,000 seconds. The default value is 300 seconds. A zero value disabled aging in the WebUI. To disable aging in the CLI, use the aging-intervale disable command.
- **beacon-interval** Sets the beacon interval. The range is 20 to 1,000 time units (1 time unit equals 1024 µs) The default value is 100 time units.
- **burst-threshold** Sets the frame burst threshold. The range is 2 to 255 frames. The default value is 3 frames.
- **cts-mode** Sets the Clear to Send (CTS) control frame protection. Does not work in 802.11b wireless mode. the default value is auto.
  - on Always use protection.
  - off Never use protection.
  - auto Automatically detects the CTS mode.
- **cts-rate** Sets the rate at which CTS frames are sent, in MBPS. Does not work in 802.11b wireless mode. The default is 11 Mbps.
- **cts-type** Sets the CTS protection type. Does not work in 802.11b wireless mode. The default is cts-only.
  - cts-only Single, self-directed frame.
  - cts-rts Two-frame exchange occurs prior to the actual network transmission.
- **dtim-period** Sets the beacon intervals between data beacon rates, referred to as DTIM. Range is 1 to 255. The default value is 1 beacon interval.

- **fragment-threshold** sets the fragmentation threshold. Range is even numbers between 256 and 2346. The default value is 2346.
- **long-preamble** Allows long preambles (802.11b wireless mode only). Default is short.
- **rts-threshold** Sets the threshold for Request to Send (RTS) packets. The range is 256 to 2346.
- **slot-time** long Disables the use of short slots. Does not work in 802.11b wireless mode. Default is short.

antenna

set wlan antenna { a | b | diversity }
unset wlan antenna

antenna	Selects a specific antenna or enables antenna diversity. Default setting is diversity. The antenna a is located closest to the power connection.

channel

set wlan channel
unset wlan channel

channel	Sets the channel for the wireless interface radio. The channel range is from 1 to 11, which is dependent on the country code and extended channel selections. Channels 12 and 13 are reserved for non-U.S. frequency regulations. Default is automatic channel selection.

country-code

set wlan country-code [string]

country-code	(This keyword is not available in the United States or Japan.) Selects a country code for which a wireless interface is configured. This setting affects the range of selectable channels and the trasmit power leve. If your region code is FCC or TELEC, you cannot set the country code.

find-channel

exec wlan find-channel

find-channel	Finds the best radio channel for the device to use for transmission.

mode

set wlan mode { 11b | 11g [11g-only] }

mode — Sets the operation mode for the wireless interface.

- 11b Allows 802.11b wireless clients to connect to the security device.
- 11g Allows 802.11b and 802.11g wireless clients to connect to the security device. The 11g-only mode allows only 802.11g wireless clients to connect to the security device.

reactivate

exec wlan reactivate

reactivate	Reboots the wireless interfaces in order for the new configurations to take effect. This command should be issued after all wireless configurations are complete.

site-survey

exec wlan site-survey

site-survey	The security device scans all channels and reports all operating wireless interfaces on the device.

transmit

set wlan transmit { power {...} | rate {...} }

transmit — Adjusts the trasmission power and rate for the wireless interface.

- power Sets the power transmission and adjusts the radio range when using more than one wireless interface in the same location and frequency. You can set the power level to an eigth, full, half, minimum, or quarter. The default is full power.
- rate Sets the wireless interface data transimission rate for sending frames. If you select auto, the wireless interface uses the best rate first, and then automatically falls back to the next rate if transmission fails. You can set 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54, or auto as the rate setting. Default is auto.

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 through 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. XAuth requires each user to enter information unique to that user (the admin name 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 security device to use Challenge Handshake Authentication Protocol (CHAP) while performing authentication with the XAuth client.
  - **query-config** Queries client settings (such as IP addresses for XAuth clients and DNS server IP addresses) from an 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 pool of IP addresses from which the security device draws when assigning addresses to XAuth clients.
- **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 a client.

Example: The following command specifies a maximum XAuth session length of 30 minutes:

set xauth lifetime 30

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 “Interface Names” on page A-I. For information on security zones, see “Zone Names” on page B-I.

Syntax

get

get zone
  [
  id *id_num* |
  all |
  zone [screen [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 | zone-based] |
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 } |
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 [dst-ip *ip_addr* | threshold *number*] |
unknown-protocol |
winnuke
}
reassembly-for-alg |
tcp-rst |
vrouter *name_str*
} |
}

Keywords and Variables

Variable Parameters

get zone *zone* [...]
set zone *zone* { ... }
unset zone *zone* { ... }

zone	The name of the zone. For more information on zones and zone names, see “Zone Names” on page B-I.

all

get zone all [...]

all	Displays information on 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 for 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 specify *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.
- **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.

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.

  The **drop-no-rpf-route** option instructs the security device to drop any packet with a source address 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 (nonroutable, as with 127.0.0.1).

  Conversely, the device does not drop the packet if the routing table contains a reverse path forwarding route that matches the source IP address on the packet. For example, the device drops an incoming packet with source IP address 10.5.1.5, if the device receives the packet on ethernet1, and there is no reverse path route for 10.5.1.5 (such as 0.0.0.0/0 or 10.5.1.0/24) on that interface. This is true even if such a reverse path exists on another interface.

  The **zone-based** option instructs the security device to base spoofing decisions on zones, instead of on individual interfaces. Enabling this setting allows sessions to continue when the device asymmetrically routes traffic between multiple interfaces in the same zone. Thus, the user can specify spoofing decisions based on either the zone or an exact interface.

  The default behavior is to base spoofing decisions on individual interfaces. To restore the default behavior, execute the following command:

  unset zone *zone* **screen ip-spoofing zone-based**

- **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. You can set the IP Sweep threshold to a value between 1 and 1,000,000 microseconds. Each time the security device receives 10 ICMP echo requests within this interval, it flags this as an IP Sweep attack, and rejects the 11th and all further ICMP packets from that host for the remainder of the second.
- **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.
- **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 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.
- **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 at which the security device makes entries in the event alarm 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 dst-ip** *ip_addr* Enables the feature and specifies the IP address of the system that you want to protect.
- **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

Example 2: 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 nonroutable source IP address:

set zone untrust screen ip-spoofing drop-no-rpf-route

Example 3: 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 4: 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 5: 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 nonsync packets.

vrouter

set zone *zone* vrouter

vrouter	Binds the zone to a virtual router.

Creating Interfaces

Example 1: The following commands:

- Create a new Layer 2 zone named *L2-Marketing* with VLAN ID number 1
- Assign physical interface *ethernet7* to the zone

set zone name L2-Marketing L2 1
set interface ethernet7 zone L2-Marketing

Example2 : The following commands:

- 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

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

```
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
    }
```

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_addr1
    [ outgoing-interface interface [ local-address ip_addr2 ] ]
      {
      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 }
          ]
      }
```

Keywords and Variables

Variable Parameters

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-1** Specifies 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. **Note:** When you omit the **auth** keyword, the device automatically uses the **null** switch. This is not advisable, because it may leave IPSec vulnerable to attack. **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

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* [local-address *ip_addr*] { ... }

outgoing-interface	Defines the interface through which the security device sends traffic for this Manual Key VPN. The **local-address** *ip_addr* value specifies the IP address of the outgoing interface for reverence by external devices. For more information on interfaces, see "Interface Names" on page A-I.

Example: The following command uses a manual tunnel.

- External gateway device IP address 1.1.1.1
- Ethernet1 as the outgoing interface, identified to outside hosts as IP address 2.2.2.2
- Specified encryption algorithm 3DES
- Password "swordfish"

set vpn tun1 manual 20001 20022 gateway 1.1.1.1 outgoing-interface ethernet1 local-address 2.2.2.2 esp 3des password swordfish

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 IP address that sends and receives traffic through the tunnel. ■ **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: The following command creates a VPN proxy configuration for a VPN (*Sales*) with the HTTP service:

set vpn Sales proxy-id local-ip 10.1.1.0/24 remote-ip 10.2.2.0/24 HTTP

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. The standard proposal provides settings recommended by Juniper Networks.

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 a 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 be placed in a VPN group or removed from it.

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 the 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 to 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 ScreenOS-specific MIB extension files into the application. The MIB extension files are on the 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

To set protocol-specific parameters, see “interface” on page 249. Protocol-specific commands for RIP, OSPF, IGMP, and PIM are listed alphabetically in this document.

Syntax

clear

clear vrouter *vrouter*
{
mroute { all | mgroup *ip_addr* [source *ip_addr*] [iif *interface*] |
protocol bgp neighbor *ip_addr* { soft-in | soft-out }
statistics
}

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 |
    mcore [ cachemiss ] |
    mroute [ brief ]
      [
      mgroup ip_addr brief |
      source ip_addr
        [
        brief |
        iif interface
        ]
      ]
    preference |
    protocol { bgp | ospf | rip | pim} |
```

NOTE: For more information on the **protocol { bgp | ospf | rip }** options, see the **bgp**, **ospf**, and **rip** command descriptions.

```
    route
      [
      id id_num |
      ip ip_addr |
      prefix ip_addr/mask |
      protocol { bgp | connected |discovered | imported | ospf | rip | static }
      source
        [ ip_addr
          [ interface interface [ gateway ip_addr ] | vrouter vrouter ]
        ]
        [
        id id_num |
        in-interface [ interface ] |
        ip ip_addr |
        prefix ip_addr/mask |
        ] |
      summary
      ] |
    route-lookup preference |
    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
      { permit | deny } { ip ip_addr/mask | default-route } number |
    add-default-route vrouter untrust-vr |
    adv-inact-interface
    auto-route-export |
    default-vrouter |
    export-to | import-from
      vrouter name_str route-map name_str protocol
        { bgp | connected | discovered | imported | ospf | rip | static }
    ignore-subnet-conflict |
```

NOTE: For more information on the **protocol { bgp | ospf | rip }** options, see the **bgp**, **ospf**, and **rip** command descriptions.

```
    max-ecmp-routes number |
    max-routes number |
    mroute
    {
      max-entries number |
      mgroup ip_addr source ip_addr iif interface oif interface out-group ip_addr|
      multiple-iif-enable |
      negative-cache [ timer number ]
    }
    nsrp-config-sync |
    pbr pbr_policy_name |
    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 } |
```

NOTE: For more information on the **protocol { bgp | ospf | pim | rip }** options, see the **bgp**, **ospf**, **pim**, and **rip** command descriptions.

```
    route [ source ] [ in-interface interface ] ip_addr/mask
      {
      interface interface
        [ gateway ip_addr ] [ metric number ] [ permanent ]
        [ preference number ][ tag id_num ] |
      vrouter name_str
      } |
    route-lookup
      preference
      [
```

```
    destination-routing number |
    sibr-routing number |
    source-routing number |
    ] |
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 |
        origin { igp | incomplete }
        preserve preference |
        preserve metric |
        tag { number | ip_addr } |
        weight number
        ]
router-id { id_num | ip_addr } |
sharable |
sibr-routing enable |
snmp trap private |
source-routing enable
]
```

Keywords and Variables

Variable Parameter

clear vrouter *vrouter* protocol bgp neighbor *ip_addr* soft-out
set vrouter *name_str*

ip_addr	Specifies an IPv4 address of BGP neighbor.
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 BGP routing context, and execute the context-dependent command **get config**.

set vrouter trust-vr
device(trust-vr)-> **set protocol bgp**
device(trust-vr/bgp)-> **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* }
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 *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.

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 on Route mode and NAT mode, refer to the *Concepts & 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.

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.
- **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-ecmp-routes

set vrouter *name_str* max-ecmp-routes *number*
unset vrouter *name_str* max-ecmp-routes

max-ecmp-routes	Specifies the maximum number of equal cost multipath (ECMP) routes to the same destination network. Enter a value between 1 and 4 (1 is the default).

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.

mcore

get vrouter *name_str* mcore [cachemiss]

mcore	Displays multicast routing information for each interface on which a multicast routing protocol is enabled.
cachemiss	Displays the current multicast cachemiss data.

mroute

get vrouter *name_str* brief
get vrouter *name_str* mroute mgroup *ip_addr1* brief
get {...} mroute mgroup *ip_addr1* source *ip_addr2* [brief | iif *interface1*]
set vrouter *name_str* mroute max-entries *number*
set {...} mroute mgroup *ip_addr1* source *ip_addr2* iif *interface1* oif *interface2*
set {...} mroute mgroup *ip_addr1* {...} out-group *ip_addr3*
set vrouter *name_str* mroute multiple-iif-enable
set vrouter *name_str* negative-cache [timer *number*]
unset vrouter *name_str* mroute max-entries
unset {...} mroute mgroup *ip_addr1* source *ip_addr2* iif *interface1* oif *interface2*
unset vrouter *name_str* mroute multiple-iif-enable
unset vrouter *name_str* negative-cache [timer *number*]

brief	Displays summary information.
max-entries	Specifies the maximum number of multicast routes allowed in the multicast routing table.
mroute	Configures a static multicast route in the specified virtual router. ■ *ip_addr1* is the multicast group address of the route ■ *ip_addr2* is the source address of the multicast data ■ *interface1* is the incoming interface of the multicast data ■ *interface2* is the outgoing interface of the multicast data ■ *ip_addr3* is the multicast group address on the outgoing interface
multiple-iif-enable	Permits multiple multicast routes for the same source and group.
negative-cache	Creates negative multicast routes if the protocol that owns the interface on which the packet was received cannot create a forwarding multicast route. The security device drops packets when they need to go on a negative multicast route. You can also set the timer value to specify the duration, in seconds, that the security device maintains the entries in the negative cache. The security device removes the entry in the negative cache when it receives information enabling it to create a forwarding multicast route entry.

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.

pbr

set vrouter *name_str* pbr *pbr_policy_name*
unset vrouter *name_str* pbr *pbr_policy_name*

pbr	Binds a Policy Based Routing (PBR) policy to the specified virtual router (VR). The PBR policy bound to the VR is used all PBR-enabled interfaces belonging to that VR. No PBR policy is solely bound at the interface level or at the zone level of an interface. For more information about PBR, see “match-group” and “pbr access-list” and “action-group”

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.
- **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 | pim }
set vrouter *name_str* protocol { bgp | ospf | rip | pim }
unset vrouter *name_str* protocol { bgp | ospf | rip | pim }

protocol

Places the security device in the context of the specified protocol : BGP, OSPF, RIP or PIM. (For information on these contexts, see the **bgp**, **ospf**, **rip** or **pim** 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] [in-interface *interface*] *ip_addr/mask* [...]
unset vrouter *name_str* route [source] [in-interface *interface*] *ip_addr/mask*
[...]

route

Configures routes for the routing table for the virtual router.

- *ip_addr/mask* Specifies the IP address that appears in the routing table.
- **gateway** *ip_addr* Specifies the gateway for the next hop.
- **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.
- **in-interface** *interface* For source interface-based routes, specifies the interface on which a packet arrives on the security device. You can then forward that traffic to either a routed interface or to a virtual router.
- **interface** *interface* Specifies the interface on which a packet for this route is to be forwarded.
- **ip** *ip_addr* Displays the route for the specified IP address.
- **metric** *number* Specifies the cost of the route. Specify a value between 1 and 65535.
- **permanent** Specifies that the route is kept active when the interface is down or the IP address is removed from the interface.
- **preference** *number* Specifies the preference value for the route. Specify a value between 0 and 255.
- **prefix** *ip_addr/mask* Displays the routes within the specified subnet address.
- **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*
  - *ip_addr/netmask* **interface** *interface* **gateway** *ip_addr* sets a gateway as the next hop.
  - **vrouter** *vrouter* sets a virtual router as the next hop.
- **summary** Displays a summary of the routes.
- **tag** *number* For destination-based routes, specifies the tag for this route. The tag can be used as a filter when redistributing routes (see the **route-map** keyword). Specify a value between 1 and 65535.
- **vrouter** *name_str* Specifies a virtual router as the next hop.

Example 1: This example sets a source based route. Traffic enters at ethernet1/1, and the next hop is set to be the virtual router *untrust-vr*.

set vrouter trust-vr route source ethernet1/1 10.2.2.1/24 vrouter untrust-vr

Example 2: This example sets a source interface-based route (SIBR). Traffic enters at ethernet1/1, and the next hop is set to be the virtual router *untrust-vr*.

set vrouter trust-vr route source in-interface ethernet1/1 10.2.2.1/24 vrouter untrust-vr

route-lookup preference

get vrouter *name_str* route-lookup preference
set vrouter *name_str* route-lookup preference [destination-routing *number*] [sibr-routing *number*] [source-routing *number*]
unset vrouter *name_str* route-lookup preference

route-lookup preference

Configures the order in which route lookups occur in the virtual router. The route lookup type that has the highest preference value is performed first, followed by the next highest preference value. The route lookup type that has the lowest preference value is performed last. Enter a number between 1-255 for the preference.

- **destination-routing** *number* Specifies the preference for route lookups based on destination IP address. The default value is 1.
- **sibr-routing** *number* Specifies the preference for route lookups based on source interface. The default value is 3.
- **source-routing** *number* Specifies the preference for route lookups based on source IP address. The default is 2.

route-map

get vrouter *name_str* route-map [...]
set vrouter *name_str* { ... } vrouter *name_str* route-map
{ name *name_str* | *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 **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 value to increment the metric for the matching route. For RIP routes, you can use this option for routes that are advertised or routes that are learned. For other routes, you can use this option to routes that are exported into another virtual router.

- **origin** Specifies the origin of a route advertised by BGP
- **preserve metric** Specifies that the metric value for the matching route is preserved when the route is exported to another virtual router.
- **preserve preference** Specifies that the preference value for the matching route is preserved when the route is exported to another virtual router.
- **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.

sibr-routing enable

set vrouter *name_str* sibr-routing enable
unset vrouter *name_str* sibr-routing enable

source- routing enable	Directs the virtual router to perform routing table lookups based on the source interface.

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.)

soft-in

clear vrouter *trust-vr* protocol bgp neighbor *ip_addr* soft-in

soft-in	Enables a soft reset and generates an inbound update from a BGP neighbor. A soft reset allows the application of a new or changed policy without clearing an active BGP session. The route-refresh feature occurs on a per-neighbor basis and does not require preconfiguration or extra memory.

soft-out

clear vrouter *trust-vr* protocol bgp neighbor *ip_addr* soft-out

soft-out	Enables a soft reset and sends a new set of updates to a BGP neighbor. A soft reset allows the application of a new or changed policy without clearing an active BGP session. The route-refresh feature occurs on a per-neighbor basis; and outbound resets don't require preconfiguration or routing table update storage.

source-routing enable

set vrouter *name_str* source-routing enable
unset vrouter *name_str* source-routing enable

source-routing enable	Directs the virtual router to perform routing table lookups based on source IP address.

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 a virtual system (vsys) from the root level of a security device.

A vsys allows you to logically partition a single security system to provide multi-tenant services. Each vsys is a unique security domain and can have its own administrators, known as *virtual system administrators* or *vsys admins*. Such adminstrators 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

Variable Parameters

get vsys [*name_str*]
set vsys *name_str*
unset vsys *name_str*

name_str	Defines the name of a virtual system (vsys) and automatically places the root level admin within the vsys. Subsequent commands configure the newly created vsys.

Example: The following command creates a virtual system named *vsys1* and switches the console to the new virtual system:

device-> **set vsys vsys1**
device(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 *name_str* for the virtual router or the nsrp vsd number.
  - **id** *id_num* Assigns an identification number to the virtual router.
  - **vsd** *number* See vsd on page 673.
- **share** Specifies a shared root-level virtual router to use as a default router for a specified vsys with name *name_str* or nsrp vsd *number*.
- **vsd** *number* See vsd on page 673.

Example 1: The following command creates a vsys named *Acme_Org*, creates a virtual router named *Acme_Router* with vsd number *3*, and switches the console to the new virtual system:

set vsys Acme_Org vrouter name Acme_Router vsd 3

Example 2: The following command creates a vsys 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. The VSD number can be 1 through 8. 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 on VSD groups, refer to the *Concepts & Examples ScreenOS Reference Guide*.

Example: The following command creates a vsys named *Acme_Org*, creates a virtual router named *Acme_Router*, creates a VSD number *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

Variable Parameters

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 Web authentication (WebAuth).

The WebAuth authentication method requires that a user first initiate a HyperText Transfer Protocol (HTTP) session and provide authentication information before being allowed to send traffic to the destination node.

You specify authentication settings in policy definitions (see “auth” on page 55).

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 *wa_serv1*:

set webauth server wa_serv1

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 host name.

port

set webtrends port *port_num*
unset webtrends port

port *port_num*	Specifies the WebTrends host port.

wlan

Use the **wlan** commands to configure the wireless local area network (WLAN) features.

Syntax

exec

```
exec wlan { find-channel | reactivate | site-survey }
```

get

```
get wlan [ acl ]
```

set

```
set wlan { 0 | 1 }
    {
    acl { mac_addr { allow | deny } | mode { enable | strict } } |
    advanced
      { aging-interval { disable | number} |
      beacon-interval { number } |
      burst-threshold { number } |
      cts-mode { auto | off | on } |
      cts-rate { 1 | 11 | 2 | 5.5 } |
      cts-type { cts-only | cts-rts } |
      dtim-period { number } |
      fragment-threshold { number } |
      long-preamble |
      rts-threshold { number } |
      slot-time long } |
    antenna { a | b | diversity } |
    channel { auto | number } |
    country-code { name_str } |
    extended-channel
    mode { 11b | 11g [ 11g-only ] | 11a | turbo } |
    super-g |
    transmit { power { eighth | full | half | minimum | quarter } |
            rate
              {
              auto | 0.25 | 0.5 | 1 | 2 | 3 | 5.5 | 11 | 6 | 9 | 12 | 18 | 24 | 36 | 48 | 54
              | 72 | 96 | 108
              }
            }
    wmm { ap { 0 | 1 | 2 | 3 } | enable | sta { 0 | 1 | 2 | 3 } }
    xr
    }
```

Keywords and Variables

0 | 1

get wlan {0 | 1 } {...}
set wlan {0 | 1 } {...}
unset wlan {0 | 1 } {...}

0 | 1	For security devices with two radio transceivers, you must specify which WLAN you are configuring. ■ 0: 2.4 GHz radio band ■ 1: 5 GHz radio band When configuring security devices with only one radio transceiver, you do not need specify the WLAN (0 or 1) when using the **wlan** commands.

acl

get wlan acl
set wlan acl { *mac_addr* { allow | deny } | mode { disable | enable | strict } }
unset wlan acl { *mac_addr* | mode }

acl	Allows or denies network access to stations with the specified MAC address (*mac_addr*). You can specify a maximum of 128 MAC addresses.
mode	Sets the wireless client restriction: ■ **enable**: Wireless clients that match the deny list are not allowed. All other clients are allowed. ■ **strict**: Wireless clients that match the allow list are allowed. All other clients are denied.

Example: The following commands set the WLAN to allow only the wireless client with MAC address 000bdfd781f9 to access the security device:

set wlan acl mode strict
set wlan acl 000bdfd781f9 allow

advanced

set wlan advanced { ... }
unset wlan advanced { ... }

advanced	Allows you to configure the following advanced WLAN settings. ■ **aging-interval** Specifies the amount of time that elapses before a wireless client is disconnected if there is no traffic to or from the client. After the aging-interval elapses and a client is disconnected, its MAC information is deleted from a MAC table on the security device. The MAC table for each radio can contain up to 60 client MAC addresses. Because new clients are denied connectivity when the MAC table is full, set the aging-interval so that existing clients whose connections are not being used are disconnected and their MAC addresses are removed from the MAC table in a timely manner. The value range is 60 through 1,000,000 seconds. The default value is 300 seconds. To disable aging, use the **aging-interval disable** command.

- **beacon-interval** Sets the interval at which beacons are sent. The value range is 20 to 1,000 time units (1 time unit equals 1024 µs) The default value is 100 time units.
- **burst-threshold** Sets the frame burst threshold. The range is 2 to 255 frames. The default value is 3 frames.
- **cts-mode** Sets the Clear to Send (CTS) control frame protection. Does not work in 802.11b wireless mode. The default value is auto.
  - **on** Always use protection.
  - **off** Never use protection.
  - **auto** Automatically detects the CTS mode.
- **cts-rate** Sets the rate at which CTS frames are sent, in Mbps. Does not work in 802.11b wireless mode. Valid values are 1, 2, 5.5, and 11 Mbps. The default is 11 Mbps.
- **cts-type** Sets the CTS protection type. Does not work in 802.11b wireless mode. The default is cts-only.
  - **cts-only** Single, self-directed frame.
  - **cts-rts** Two-frame exchange occurs prior to the actual network transmission.
- **dtim-period** Sets the number of beacons that are sent before the delivery traffic indication map (DTIM) is sent. Increasing the DTIM period decreases the number of broadcasts sent to clients. Range is 1 to 255. The default value is 1 beacon interval.
- **fragment-threshold** Sets the maximum length of a frame before it is fragmented into multiple frames before transmission. Value range is even numbers between 256 and 2346. The default value is 2346.
- **long-preamble** Allows use of long preambles (802.11b and 802.11g wireless mode only). Default is short.
- **rts-threshold** Sets the maximum length a frame is before using the Request to Send (RTS) method to send the frame. The range is 256 to 2346.
- **slot-time long** Enables use of long slot time. Used only for 802.11g. Default is short.

antenna

set wlan antenna { a | b | diversity }
unset wlan antenna

antenna	Selects a specific antenna to be used or enables antenna diversity. Default setting is diversity. For information about antennae, see the hardware manual for your security device. ■ **a**: Uses antenna A ■ **b**: Uses antenna B ■ **diversity**: Uses antenna A or B, whichever antenna has the stronger signal

channel

set wlan channel
unset wlan channel

channel	Sets the channel for the wireless interface radio. The channel range is 1 through 11 and is dependent on the country code and extended channel selections. Channels 12 and 13 are reserved for non-U.S. frequency regulations. Default is automatic channel selection.

country-code

set wlan country-code { string }

country-code	(This keyword is not available in the United States or Japan.) Defines the country in which the security device is operating. This setting determines the channels and the transmit power level you can configure. If your region code is FCC or TELEC, you cannot set the country code. For a list of country codes, see the *Concepts & Examples ScreenOS Reference Guide*.

extended-channel

set wlan extended-channel

extended-channel	For the 2.4 GHz radio band, enables use of channels 12 and 13 if the regulatory domain allows the use of these channels. Although enabling extended-channel mode provides better geographic coverage, the data throughput rate for clients might be decreased.

find-channel

exec wlan find-channel

find-channel	Finds the best radio channel for the device to use for transmission. Use this command if you do not want to use the auto keyword to automatically select channels and want to find the channel with the least interference.

mode

set wlan mode { 11a | 11b | 11g [11g-only] | turbo }
unset wlan mode

mode — Sets the operation mode for the wireless interface.

- **11a**: Allows 802.11a wireless clients to connect to the security device.
- **11b** Allows 802.11b wireless clients to connect to the security device.
- **11g** Allows 802.11b and 802.11g wireless clients to connect to the security device. The 11g-only mode allows only 802.11g wireless clients to connect to the security device.
- **turbo**: Enables static turbo mode for 2.4 GHz and 5 GHz radio bands. Turbo mode allows data transmit rate of up to 108 Mbps.

  If you enable turbo mode, wireless clients must also support turbo mode. If wireless clients do not support turbo mode, they cannot connect to the wireless network.

reactivate

exec wlan reactivate

reactivate	Reboots the wireless interfaces so that the new configurations take effect. Use this command after all wireless configurations are complete. Depending on your network, rebooting the wireless interfaces can take 60 seconds or more. Wireless traffic is disrupted, and all wireless client sessions are terminated.

site-survey

exec wlan site-survey

site-survey	The security device scans all channels and reports all access points in the surrounding area. Use this command to find rogue access points. Depending on your network, the site survey can take approximately 60 seconds and disrupts wireless network traffic.

super-g

set wlan super-g
unset wlan super-g

super-g

Enables the Atheros Super G feature, which can increase user data throughput rate up to 4 Mbps for 802.11a and 802.11g clients by using the following methods:

- Bursting: Allows the device to transmit multiple frames in a burst rather than pausing after each frame.
- Fast frames: Allows for more information per frame to be transmitted by allowing a larger-than-standard frame size.
- Compression: Link-level hardware compression is performed by a built-in data compression engine.

If wireless clients do not support Super G and the security device has Super G enabled, they can still connect to the wireless network, but the Super G feature is not available.

transmit

set wlan transmit { power {...} | rate {...} }
unset wlan transmit { power | rate }

transmit

Adjusts the transmission power and rate for the wireless interface.

- **power** Sets the power transmission and adjusts the radio range. You can set the power level to an eighth, full, half, minimum, or quarter of maximum transmit power, which is the maximum power allowed in the country the security device is operating in. The default is full power.
- **rate** Sets the minimum data transmit rate in megabits per second (Mbps) for sending frames. The data transmit rate depends on the radio type.
  - 802.11a: 6, 9, 12, 18, 24, 36, 48, 54
  - 802.11a with XR enabled: 0.25, 0.5, 3, 6, 9, 12, 18, 24, 36, 48, 54
  - 802.11b: 1, 2, 5.5, 11
  - 802.11g: 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54
  - 802.11g with XR enabled: .0.25, 0.5, 1, 2, 3, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54
  - If turbo is enable: 12, 18, 24, 36, 48, 72, 96, 108

  The **auto** rate, which is the default value, uses the best rate first and then automatically falls back to the next rate if transmission fails.

wmm

set wlan { 0 | 1 } wmm { ap {...} | enable | sta {... } }
unset wlan { 0 | 1 } wmm { ... }

ap

Configures Wi-Fi Multimedia (WWM) on the access point (security device) side of the wireless connection. You can set WWM parameters for the following categories:

- 0 (Best effort)
- 1 (Background)
- 2 (Video)
- 3 (Voice)

For each category, you can set the following parameters:

- Logcwmin and logcwmax: WMM defines a Contention Window (CW), which is equivalent to a random backoff period.

  The CWmin parameter specifies the minimum number of slots of the contention window used by the security device or client for a particular AC to generate a random number for the backoff. If logcwmin is x, then CWmin is 2^x-1.

  The CWmax parameter specifies the maximum number of slots of the window used by the security device or client for a particular AC to generate a random number for the backoff. If logcwmax is x, then CWmax is 2^x-1.
- Aifs: Arbitrary Inter-Frame Space Number (AIFSN) specifies the number of slots, after a SIFS duration, that the security device or client for an AC will check the medium-idle before transmitting or executing a backoff.
- Txoplimit: Transmit Opportunity specifies the maximum amount of time the security device or client can initiate transmissions. If you set txoplimit to x, the maximum time is 32*x microseconds.
- Acm: Admission Control is an optional feature and is not currently supported.
- Ack Policy: You can enable or disable an acknowledgement policy for a WAP. This parameter does not apply to clients.

enable

Enables WMM.

sta — Configures WWM on the station side of the wireless connection. You can set WWM parameters for the following access categories (ACs):

- 0 (Best effort)
- 1 (Background)
- 2 (Video)
- 3 (Voice)

For each category, you can set the following parameters:

- Logcwmin and logcwmax: WMM defines a Contention Window (CW), which is equivalent to a random backoff period.

  The CWmin parameter specifies the minimum number of slots of the contention window used by the security device or client for a particular AC to generate a random number for the backoff. If logcwmin is x, then CWmin is 2^x-1.

  The CWmax parameter specifies the maximum number of slots of the window used by the security device or client for a particular AC to generate a random number for the backoff. If logcwmax is x, then CWmax is 2^x-1.
- Aifs: Arbitrary Inter-Frame Space Number (AIFSN) specifies the number of slots, after a SIFS duration, that the security device or client for an AC will check the medium-idle before transmitting or executing a backoff.
- Txoplimit: Transmit Opportunity specifies the maximum amount of time the security device or client can initiate transmissions. If you set txoplimit to x, the maximum time is 32*x microseconds.
- Acm: Admission Control is an optional feature and is not currently supported.

xr

set wlan xr
unset wlan xr

xr — Enables eXtended Range (XR) technology. XR processes 802.11 signals, defined by IEEE 802.11a and 802.11g standards, so that wireless networks to have fewer “dead spots” and greater range than usual. XR processes weaker signals more effectively and allows greater coverage.

Only the first active SSID per radio can support XR. When XR is enabled, the first active SSID per radio uses the XR feature.

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 through 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. XAuth requires each user to enter information unique to that user (the admin name 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 security device to use Challenge Handshake Authentication Protocol (CHAP) while performing authentication with the XAuth client.
  - **query-config** Queries client settings (such as IP addresses for XAuth clients and DNS server IP addresses) from an 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 pool of IP addresses from which the security device draws when assigning addresses to XAuth clients.
- **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 a client.

Example: The following command specifies a maximum XAuth session length of 30 minutes:

set xauth lifetime 30

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 "Interface Names" on page A-I. For information about security zones, see "Zone Names" on page B-I.

Syntax

get

```
get zone
    [
    id id_num |
    all |
    zone [ screen [ attack | counter | info ] ]
    ]
```

set

```
set zone
    {
    name zone [ L2 id_num | tunnel zone ] |
    zone
      {
      asymmetric-vpn |
      block |
      pbr pbr_policy_name |
      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 | zone-based ] |
        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 } |
        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 [ dst-ip ip_addr | threshold number ] |
        unknown-protocol |
        winnuke
        }
      reassembly-for-alg |
      tcp-rst |
      vrouter name_str
      } |
    }
```

Keywords and Variables

Variable Parameters

get zone *zone* [...]
set zone *zone* { ... }
unset zone *zone* { ... }

zone	The name of the zone. For more information on zones and zone names, see “Zone Names” on page B-I.

all

get zone all [...]

all	Displays information on 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 for 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 specify *untrust* as the out zone:

set zone name Engineering tunnel untrust

pbr

set zone *zone* pbr *pbr_policy_name*
unset zone *zone* pbr *pbr_policy_name*

pbr	Binds a Policy Based Routing policy to the specified zone. A PBR policy bound to a zone is used by all PBR-enabled interfaces within that zone. In this case, the PBR policy is not bound at the interface level.

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.
- **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.

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.

  The **drop-no-rpf-route** option instructs the security device to drop any packet with a source address 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 (nonroutable, as with 127.0.0.1).

  Conversely, the device does not drop the packet if the routing table contains a reverse path forwarding route that matches the source IP address on the packet. For example, the device drops an incoming packet with source IP address 10.5.1.5, if the device receives the packet on ethernet1, and there is no reverse path route for 10.5.1.5 (such as 0.0.0.0/0 or 10.5.1.0/24) on that interface. This is true even if such a reverse path exists on another interface.

  The **zone-based** option instructs the security device to base spoofing decisions on zones, instead of on individual interfaces. Enabling this setting allows sessions to continue when the device asymmetrically routes traffic between multiple interfaces in the same zone. Thus, the user can specify spoofing decisions based on either the zone or an exact interface.

  The default behavior is to base spoofing decisions on individual interfaces. To restore the default behavior, execute the following command:

  unset zone *zone* **screen ip-spoofing zone-based**
- **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. You can set the IP Sweep threshold to a value between 1 and 1,000,000 microseconds. Each time the security device receives 10 ICMP echo requests within this interval, it flags this as an IP Sweep attack, and rejects the 11th and all further ICMP packets from that host for the remainder of the second.
- **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.
- **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 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.

- **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 at which the security device makes entries in the event alarm 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 dst-ip** *ip_addr* Enables the feature and specifies the IP address of the system that you want to protect.
- **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

Example 2: 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 nonroutable source IP address:

set zone untrust screen ip-spoofing drop-no-rpf-route

Example 3: 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 4: 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 5: 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 nonsync packets.

vrouter

set zone *zone* vrouter

vrouter	Binds the zone to a virtual router.

Creating Interfaces

Example 1: The following commands:

- Create a new Layer 2 zone named *L2-Marketing* with VLAN ID number 1
- Assign physical interface *ethernet7* to the zone

set zone name L2-Marketing L2 1
set interface ethernet7 zone L2-Marketing

Example2 : The following commands:

- 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
Interface Names

Most security zones exchange traffic with other zones (or with other devices) through physical interfaces or logical subinterfaces. The interface names are as follows:

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
Zone Names

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. The names of ScreenOS security zones are as follows.

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.
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.