M through R Commands

mac-list

Adds a list of MAC addresses using a first match search. This command is used by the firewall VPN client in performing MAC-based authentication.

[**no**] **mac-list** *id* **deny**|**permit** *mac macmask*

show mac-list [*id*]

clear mac-list [*id*]

Syntax Description

deny	Traffic matching **deny** is not included in the MAC list and is subjected to both authentication and authorization.
id	MAC access list number.
mac	Source MAC address in *aabbcc.ddeeff.gghhii* form.
macmask	Applies the netmask to *mac*, which is a string of 1's followed by 0's in the form *aabbcc.ddeeff.gghhii*, and allows the grouping of MAC addresses.
permit	Traffic matching **permit** is included in the MAC list and is exempt from authentication and authorization.

Defaults

None.

Command Modes

The **mac-list** command is available in configuration mode.

The **show mac-list** command is available in privileged mode.

Usage Guidelines

The **mac-list** command, similar to the **access-list** command, can be entered multiple times with same *id* to group a set of MAC addresses.

Only AAA exemption is provided. Authorization is automatically exempted for MACs for which authentication is exempted. Other types of AAA with **mac-list** are not supported.

The **clear aaa** command removes the **mac-list** command statements along with the rest of the AAA configuration.

The **show aaa** command displays **mac-list** command statements as part of the AAA configuration.

Note When configuring **mac-exempt**, it is recommended not to use the same IP address for both the MACs. However, in case the the hosts are getting their IP addresses from a DHCP Server, one can receive an IP address that another host in the same network used earlier. For example, if the **mac-exempt** command is configured for both the MACs, M1 and M2 when these two hosts are getting their IP addresses from the DHCP Server. Assume M1 with IP1 has gone through the PIX firewall earlier. At a later time, both hosts will get new IP addresses from the DHCP Server and this time M2 gets IP1. In this case the traffic from M1 is allowed to go through but the traffic from M2 would be dropped. However, If a **mac-exempt** is configured for one of them, then the traffic from both hosts would be allowed to pass in case they happend to send the traffic with the same IP address. A syslog alerting you to a possible spoof attack, is generated.

Examples

The following example shows how to configure a MAC access list:

```
pixfirewall(config)# mac-list adc permit 00a0.c95d.0282 ffff.ffff.ffff
pixfirewall(config)# mac-list adc deny 00a1.c95d.0282 ffff.ffff.ffff
pixfirewall(config)# mac-list ac permit 0050.54ff.0000 ffff.ffff.0000
pixfirewall(config)# mac-list ac deny 0061.54ff.b440 ffff.ffff.ffff
pixfirewall(config)# mac-list ac deny 0072.54ff.b440 ffff.ffff.ffff

pixfirewall(config)# show mac-list
mac-list adc permit 00a0.c95d.0282 ffff.ffff.ffff
mac-list adc deny 00a1.c95d.0282 ffff.ffff.ffff
mac-list ac permit 0050.54ff.0000 ffff.ffff.0000
mac-list ac deny 0061.54ff.b440 ffff.ffff.ffff
mac-list ac deny 0072.54ff.b440 ffff.ffff.ffff
```

Related Commands

aaa authentication	Enable, disable, or view LOCAL, TACACS+, or RADIUS user authentication on a server designated by the **aaa-server** command, or PDM user authentication.
aaa authorization	Enable or disable LOCAL or TACACS+ user authorization services.
	Exempts a list of MAC addresses from authentication and authorization.
access-list	Create an access list, or use downloadable access lists. (Downloadable access lists are supported for RADIUS servers only.)

management-access

Enables access to an internal management interface on the firewall.

[**no**] **management-access** *mgmt_if*

show management-access

Syntax Description

mgmt_if	The name of the firewall interface to be used as the internal management interface.

Defaults

None.

Command Modes

The **management-access** *mgmt_if* command is available in configuration mode.

The **show management-access** is available in privileged mode.

Usage Guidelines

The **management-access** *mgmt_if* command enables you to define an internal management interface using the IP address of the firewall interface specified in *mgmt_if*. (The firewall interface names are defined by the **nameif** command and displayed in quotes, " ", in the **show interface** output.)

In PIX Firewall software Version 6.3, this command is supported for the following through an IPSec VPN tunnel only, and only one management interface can be defined globally:

- SNMP polls to the *mgmt_if*
- HTTPS requests to the *mgmt_if*
- PDM access to the *mgmt_if*
- Telnet access to the *mgmt_if*
- SSH access to the *mgmt_if*
- Ping to the *mgmt_if*

The **show management-access** command displays the firewall management access configuration.

Examples

The following example shows how to configure a firewall interface named "inside" as the management access interface:

```
pixfirewall(config)# management-access inside
pixfirewall(config)# show management-access
management-access inside
```

mgcp

Configures additional support for the Media Gateway Control Protocol (MGCP) fixup (packet application inspection) and is used with the **fixup protocol mgcp** command.

[**no**] **mgcp call-agent** *ip_address group_id*

[**no**] **mgcp command-queue** *limit*

[**no**] **mgcp gateway** *ip_address group_id*

show mgcp {**commands** | **sessions**} [**detail**]

clear mgcp

Syntax Description

commands	The MGCP commands in the MGCP configuration on the firewall.
group_id	The ID of the Call Agent group, from 0 to 4294967295.
ip_address	The IP address of the gateway.
limit	Maximum number of commands to queue, from 1 to 4294967295.
sessions	The MGCP active sessions.

Defaults

The default for the MGCP command queue is 200.

Command Modes

The **mgcp** command is available in configuration mode.

The **show mgcp** command is available in privileged mode.

Usage Guidelines

The **mgcp** commands are used to provide additional support for the MGCP fixup. The MGCP fixup itself is enabled with the **fixup protocol mgcp** command.

mgcp call-agent

The **mgcp call-agent** command is used to specify a group of Call Agents that can manage one or more gateways. The Call Agent group information is used to open connections for the Call Agents in the group (other than the one a gateway sends a command to) so that any of the Call Agents can send the response. Call Agents with the same *group_id* belong to the same group. A Call Agent may belong to more than one group. The *group_id* option is a number from 0 to 4294967295. The *ip_address* option specifies the IP address of the Call Agent.

mgcp command-queue

The **mgcp command-queue** command specifies the maximum number of MGCP commands that are queued while waiting for a response. The range of allowed values is from 1 to 4294967295. The default is 200. When the limit has been reached and a new command arrives, the command that has been in the queue for the longest time is removed.

mgcp gateway

The **mgcp gateway** command is used to specify which group of Call Agents are managing a particular gateway. The IP address of the gateway is specified with the *ip_address* option. The *group_id* option is a number from 0 to 4294967295 that must correspond with the *group_id* of the Call Agents that are managing the gateway. A gateway may only belong to one group.

clear mgcp and show mgcp

The **clear mgcp** command removes the MGCP configuration and resets the command queue limit to the default of 200.

The **show mgcp commands** command lists the number of MGCP commands in the command queue. The **show mgcp sessions** command lists the number of existing MGCP sessions. The **detail** option includes additional information about each command (or session) in the output.

Examples

The following example limits the MGCP command queue to 150 commands, allows Call Agents 10.10.11.5 and 10.10.11.6 to control gateway 10.10.10.115, and allows Call Agents 10.10.11.7 and 10.10.11.8 to control both gateways 10.10.10.116 and 10.10.10.117:

```
pixfirewall(config)# mgcp call-agent 10.10.11.5 101
pixfirewall(config)# mgcp call-agent 10.10.11.6 101
pixfirewall(config)# mgcp call-agent 10.10.11.7 102
pixfirewall(config)# mgcp call-agent 10.10.11.8 102
pixfirewall(config)# mgcp command-queue 150
pixfirewall(config)# mgcp gateway 10.10.10.115 101
pixfirewall(config)# mgcp gateway 10.10.10.116 102
pixfirewall(config)# mgcp gateway 10.10.10.117 102
```

The following are examples of the **show mgcp** command options:

```
pixfirewall# show mgcp commands
1 in use, 1 most used, 200 maximum allowed
CRCX, gateway IP: host-pc-2, transaction ID: 2052, idle: 0:00:07
pixfirewall# show mgcp commands detail
1 in use, 1 most used, 200 maximum allowed
CRCX, idle: 0:00:10
        Gateway IP      host-pc-2
        Transaction ID  2052
        Endpoint name   aaln/1
        Call ID         9876543210abcdef
        Connection ID
        Media IP        192.168.5.7
        Media port      6058

pixfirewall# show mgcp sessions
1 in use, 1 most used
Gateway IP host-pc-2, connection ID 6789af54c9, active 0:00:11

pixfirewall# show mgcp sessions detail
1 in use, 1 most used
Session active 0:00:14
        Gateway IP      host-pc-2
        Call ID         9876543210abcdef
        Connection ID   6789af54c9
        Endpoint name   aaln/1
        Media lcl port  6166
        Media rmt IP    192.168.5.7
        Media rmt port  6058
```

Related Commands

debug	Displays debug information for Media Gateway Control Protocol (MGCP) traffic.
fixup protocol	Enables the Media Gateway Control Protocol (MGCP) fixup. Use with the **mgcp** command to configure additional support for the MGCP fixup.
show conn	Displays all active connections. There is an MGCP **show conn** option and connection flag, “g”.
timeout	Sets the maximum idle time duration. (There is an MGCP timeout option.)

mroute

Configures a static multicast route.

[**no**] **mroute** *src smask in_if_name dst dmask out_if_name*

show mroute [*dst* [*src*]]

Syntax Description

dmask	The destination network address mask.
dst	The Class D address of the multicast group.
in_if_name	The input interface name to pass multicast traffic.
out_if_name	The output interface name to pass multicast traffic.

smask	The multicast source network address mask.
src	The IP address of the multicast source.

Command Modes

Configuration mode.

Usage Guidelines

The **mroute** command supports routing multicast traffic through the PIX Firewall.

The **show mroute** command displays the current multicast route table.

Examples

In the following example, the multicast sources are the inside interface and DMZ with no internal receivers:

```
multicast interface outside
multicast interface inside
multicast interface dmz

mroute 1.1.1.1 255.255.255.255 inside 230.1.1.2 255.255.255.255 outside
mroute 2.2.2.2 255.255.255.255 dmz 230.1.1.2 255.255.255.255 outside
```

The following example shows sample output from the **show mroute** command. This output shows that the PIX Firewall has dropped 502 packets because of an empty output interface list (Olist).

```
pixfirewall(config)# show mroute
IP Multicast Forwarding Information Base
Entry flags: C - Directly-Connected Check, S - Signal, D - Drop
Interface flags: F - Forward, A - Accept, IC - Internal Copy,
NS - Negate Signal, DP - Don't Preserve, SP - Signal Present,
EG - Egress
Forwarding Counts: Packets in/Packets out/Bytes out
Failure Counts: RPF / TTL / Empty Olist / Other


(*,225.2.1.14),  Flags: S
  Last Used: 0:02:18
  Forwarding Counts: 4/1/188
  Failure Counts: 0/0/3/0
  inside Flags: F

(192.168.1.35,225.2.1.14),  Flags:
  Last Used: 17:57:09
  Forwarding Counts: 502/0/0
  Failure Counts: 0/0/502/0
  outside Flags: A SP
  inside Flags: F
```

Even though the Outside Flags indicate that the PIX Firewall is receiving the multicast traffic, and the Inside Flag indicates it is forwarding the traffic, an ACL on the Outside interface is causing the inbound multicast traffic stream to be dropped.

mtu

Specify the maximum transmission unit (MTU) for an interface.

[**no**] **mtu** *if_name bytes*

show mtu

Syntax Description

bytes	The number of bytes in the MTU, in the range of 64 to 65,535 bytes. The value specified depends on the type of network connected to the interface.
if_name	The internal or external network interface name.

Command Modes

Configuration mode.

Usage Guidelines

The **mtu** command sets the size of data sent on a connection. Data larger than the maximum transmission unit (MTU) value is fragmented before being sent. The minimum value for *bytes* is 64 and the maximum is 65,535 bytes.

For PIX Firewall software Version 6.2, MTU size must be greater than or equal to 1500 for the Stateful Failover link and greater than or equal to 576 for the LAN-based failover link.

For PIX Firewall software Versions 5.2 through 6.1, MTU size must be greater than or equal to 256 bytes for the Stateful Failover link.

PIX Firewall supports the IP Path MTU Discovery mechanism, as defined in RFC 1191. IP Path MTU Discovery allows a host to dynamically discover and cope with differences in the maximum allowable maximum transmission unit (MTU) size of the various links along the path. Sometimes a PIX Firewall is unable to forward a datagram because it requires fragmentation (the packet is larger than the MTU you set for the interface), but the "don't fragment" (DF) bit is set. The network software sends a message to the sending host, alerting it to the problem. The host will have to fragment packets for the destination so that they fit the smallest packet size of all the links along the path.

For Ethernet interfaces, the default MTU is 1500 bytes in a block, which is also the maximum. This value is sufficient for most applications, but you can pick a lower number if network conditions warrant it.

The **no mtu** command resets the MTU block size to 1500 for Ethernet interfaces. The **show mtu** command displays the current block size. The **show interface** command also shows the MTU value.

Note For the MTU fragmentation to work properly when using L2TP, we recommend that the MTU size be set to 1380, in order to account for the L2TP header and IPSec header length.

Examples

The following example shows the use of the **mtu** command with Ethernet:

```
interface ethernet1 auto
mtu inside 8192

show mtu
mtu outside 1500
mtu inside 8192
```

multicast

Enables multicast traffic to pass through the PIX Firewall. Includes an **igmp** subcommand mode for multicast support.

[**no**] **multicast interface** *interface_name*

clear multicast

show igmp [*group* | **interface** *interface_name*] [**detail**]

show multicast [**interface** *interface_name*]

Subcommands to the **multicast** command:

igmp forward interface *interface_name*

igmp access-group *id*

igmp version {**1** | **2**}

igmp join-group *group*

igmp max-groups *number*

igmp query-interval *seconds*

igmp query-max-response-time *seconds*

no igmp

clear igmp [*group* | **interface** *interface_name*]

Syntax Description

detail	Displays all information in the IGMP table.
id	Access control list ID.
group	The address of the multicast group.
igmp	Internet Group Management Protocol.
interface_name	The name of the interface on which to enable multicast traffic.
join-group	The multicast group to join.
max-groups	Specifies the maximum number of groups, from 0 to 2000. The default value is 500.
number	The maximum number of groups that can be joined.
query-interval	The query response time interval.
query-max-response-time	The maximum query response time interval.
seconds	Specifies the number of seconds to wait.

Command Modes

Configuration mode.

Usage Guidelines

The **multicast** command supports routing multicast traffic through the PIX Firewall.

The PIX Firewall **igmp** commands are subcommands of the **multicast** command.

The **clear igmp** [*group* | **interface** *interface_name*] command clears IGMP entries.

Note The PIX Firewall acts as an IGMP proxy but is not a multicast router.

The **show igmp** [*group* | **interface** *interface_name*] [**detail**] command displays the IGMP information for a multicast group, whether statically configured or dynamically created.

The **show multicast** [interface *interface_name*] command displays all or per-interface multicast settings. Also displays the IGMP configuration for any interface that is specified.

Examples

The following example shows use of the **multicast** command with corresponding **igmp** subcommands:

```
multicast interface outside
multicast interface inside
    igmp forward interface outside
    igmp join-group 224.1.1.1
```

The following is sample output from the **show igmp** command:

```
pixfirewall(config)# show igmp

  IGMP is enabled on interface inside
  Current IGMP version is 2
  IGMP query interval is 60 seconds
  IGMP querier timeout is 125 seconds
  IGMP max query response time is 10 seconds
  Last member query response interval is 1 seconds
  Inbound IGMP access group is
  IGMP activity: 0 joins, 0 leaves
  IGMP querying router is 10.1.3.1 (this system)

  IGMP Connected Group Membership
   Group Address    Interface            Uptime    Expires   Last Reported
```

name/names

Associate a name with an IP address.

[no] name *ip_address name*

[no] names

clear names

show names

Syntax Description

ip_address	The IP address of the host being named.
name	The name assigned to the IP address. Allowable characters are **a** to **z**, **A** to **Z**, **0** to **9**, a dash, and an underscore. The *name* cannot start with a number. If the name is over 63 characters long, the **name** command fails.

Command Modes

Configuration mode.

Usage Guidelines

Use the **name** command to identify a host by a text name. The names you define become like a host table local to the PIX Firewall. Because there is no connection to DNS or /etc/hosts on UNIX servers, use of this command is a mixed blessing—it makes configurations much more readable but introduces another level of abstraction to administer; not only do you have to add and delete IP addresses to your configuration as you do now, but with this command, you must ensure that the host names either match existing names or you have a map to list the differences.

The **name** command maps text strings to IP addresses. The **clear names** command clears the list of names from the PIX Firewall configuration. The **no names** command disables the use of the text names, but does not remove them from the configuration. The **show names** command lists the **name** command statements in the configuration.

Usage Notes

1. You must first use the **names** command before using the **name** command. Use the **name** command immediately after the **names** command and before you use the **write memory** command.
2. To disable displaying **name** values, use the **no names** command.
3. Only one name can be associated with an IP address.
4. Both the **name** and **names** command statements are saved in the configuration.
5. While the **name** command will let you assign a name to a network mask, no other PIX Firewall command requiring a mask will let you use the name as a mask value. For example, the following command is accepted.

```
name 255.255.255.0 class-C-mask
```

Note None of the commands in which a mask is required can process the "class-C-mask" as an accepted network mask.

Examples

In the example that follows, the **names** command enables use of the **name** command. The **name** command substitutes **pix_inside** for references to 192.168.42.3, and **pix_outside** for 209.165.201.3. The **ip address** commands use these names while assigning IP addresses to the network interfaces. The **no names** command disables the **name** command values from displaying. Subsequent use of the **names** command restores their display.

```
pixfirewall(config)# names
pixfirewall(config)# name 192.168.42.3 pix_inside
pixfirewall(config)# name 209.165.201.3 pix_outside
pixfirewall(config)# ip address inside pix_inside 255.255.255.0
pixfirewall(config)# ip address outside pix_outside 255.255.255.224

pixfirewall(config)# show ip address
System IP Addresses:
    inside ip address pix_inside mask 255.255.255.0
    outside ip address pix_outside mask 255.255.255.224

pixfirewall(config)# no names
pixfirewall(config)# show ip address
System IP Addresses:
    inside ip address 192.168.42.3 mask 255.255.255.0
    outside ip address 209.165.201.3 mask 255.255.255.224

pixfirewall(config)# names
pixfirewall(config)# show ip address
System IP Addresses:
    inside ip address pix_inside mask 255.255.255.0
    outside ip address pix_outside mask 255.255.255.224

pixfirewall(config)# show names
System IP Addresses:
    name 192.168.42.3 pix_inside
    name 209.165.201.3 pix_outside
```

nameif

Name interfaces and assign security level.

nameif {*hardware_id* | *vlan_id*} *if_name security_level*

clear nameif

show nameif

Syntax Description

hardware_id	The hardware name for the network interface that specifies the interface's slot location on the PIX Firewall motherboard. For more information on PIX Firewall hardware configuration, refer to the *Cisco PIX Firewall Hardware Installation Guide*. A logical choice for an Ethernet interface is **ethernet***n*. These names can also be abbreviated with any leading characters in the name, for example, **ether1** or **e2**.
if_name	A name for the internal or external network interface of up to 48 characters in length. By default, PIX Firewall names the inside interface "inside," the outside interface "outside," and any perimeter interface "intf*n*" where *n* is 2 through 5.
security_level	Enter **0** for the outside network or **100** for the inside network. Perimeter interfaces can use any number between **1** and **99**. By default, PIX Firewall sets the security level for the inside interface to **security100** and the outside interface to **security0**. The first perimeter interface is initially set to **security10**, the second to **security15**, the third to **security20**, and the fourth perimeter interface to **security25** (a total of 6 interfaces are permitted, with a total of 4 perimeter interfaces permitted). The word **security** in this command can also be abbreviated as **sec**, for example **sec10**. For access from a higher security to a lower security level, **nat** and **global** commands or **static** commands must be present. For access from a lower security level to a higher security level, **static** and **access-list** commands must be present. Interfaces with the same security level cannot communicate with each other. We recommend that every interface have a unique security level.
vlan_id	The VLAN identifier. For example: vlan10, vlan20, etc. (*vlan_id* is configured with the **interface** command.)

Command Modes

Configuration mode.

Usage Guidelines

The **nameif** command lets you assign a name to an interface. You can use this command to assign interface names if you have more than two network interface circuit boards in your PIX Firewall. The first two interfaces have the default names **inside** and **outside**. The **inside** interface has a default security level of 100, the **outside** interface has a default security level of 0. The **clear nameif** command reverts **nameif** command statements to default interface names and security levels.

Use **nameif** *hardware_id if_name security_level* to set name of a physical interface and use the **nameif** *vlan_id if_name security_level* command to set the name of a logical interface. Physical interfaces are one per each NIC, in place at boot time, and non-removable. Logical interfaces can be many-to-one for each NIC, are created at run time, and can be removed through software reconfiguration.

Usage Notes

1. If you change the *hardware_id* of the outside interface; for example, from ethernet0 to ethernet1, PIX Firewall changes every reference to the outside interface in your configuration to inside, which can cause problems with **route**, **ip**, and other command statements that affect the flow of traffic through the PIX Firewall.
2. After changing a **nameif** command, use the **clear xlate** command.
3. The inside interface cannot be renamed or given a different security level. The outside interface can be renamed, but not given a different security level.
4. An interface is always "external" with respect to another interface that has a higher security level.

Examples

The following example shows how to use the **nameif** *hardware_id if_name security_level* command:

```
nameif ethernet2 perimeter1 sec50
nameif ethernet3 perimeter2 sec20

The following example shows how to use the nameif vlan_id if_name security_level command:
nameif vlan10 perimeter3 sec10

The following example is a configuration that uses both physical and VLAN interfaces:
nameif ethernet0 outside security0
nameif ethernet1 intf6 security90
nameif ethernet2 dmz security50
nameif vlan4 intf4 security10
nameif vlan5 intf5 security10
nameif vlan10 intf5 security10
```

Related Commands

interface	Sets network interface parameters and configures VLANs.

nat

Associate a network with a pool of global IP addresses.

[**no**] **nat** [(*local_interface*)] *id local_ip* [*mask* [**dns**] [**outside** | [**norandomseq**] [*max_conns* [*emb_limit*]]]]

[**no**] **nat** [(*local_interface*)] *id* **access-list** *acl_name* [**dns**] [**outside** | [**norandomseq**] [*max_conns* [*emb_limit*]]]

[**no**] **nat** [(*local_interface*)] **0 access-list** *acl_name* [**outside**]

clear nat

show nat

Syntax Description

access-list	Lets you identify local traffic for network address translation (NAT) by specifying the local and destination addresses (or ports). This feature is known as policy NAT. **Note** Use NAT exemption (**nat 0 access-list)** with the ACL deny statement but not with policy NAT. Use port selectors with policy NAT but not with NAT Excemption. You can only include **permit** statements in the access list. Local traffic is matched to the first matching policy NAT statement. See the "Order of NAT Commands Used to Match Local Addresses" section on page 7-19 for more information.
acl_id	Specifies the access list name.
clear nat	Removes **nat** command statements from the configuration.
dns	Specifies to use the created translation to rewrite the DNS address record.
emb_limit	Specifies the maximum number of embryonic connections per host. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination. Set a small value for slower systems, and a higher value for faster systems. The default is 0, which means unlimited embryonic connections. The embryonic connection limit lets you prevent a type of attack where processes are started without being completed. When the embryonic limit has been surpassed, the TCP intercept feature intercepts TCP synchronization (SYN) packets from clients to servers on a higher security level. The software establishes a connection with the client on behalf of the destination server, and if successful, establishes the connection with the server on behalf of the client and combines the two half-connections together transparently. Thus, connection attempts from unreachable hosts will never reach the server. The PIX firewall accomplishes TCP intercept functionality using SYN cookies. **Note** This option does not apply to outside NAT. The TCP intercept feature only applies to hosts or servers on a higher security level. If you set the *emb_limit* as well as the **outside** option, the *emb_limit* is ignored.
(*local_interface*)	Specifies the name of the network interface, as defined by the **nameif** command, through which the hosts or network designated by *local_ip* or **access-list** *acl_id* are accessed. You must enter the interface name in parentheses. If you do not enter the interface name, then the default is **inside**.
local_ip	Specifies the addresses to translate. You can use **0.0.0.0** (or **0** for short) to identify all hosts. Local traffic is matched to a **nat** statement using the best match. See the "Order of NAT Commands Used to Match Local Addresses" section on page 7-19 for more information.
mask	Specifies the IP netmask to apply to *local_ip*. If you do not specify a mask, the PIX Firewall derives the network mask from the class of the IP address. For example, the command **nat 0 10.130.36.0** causes all addresses in the 10.0.0.0 network to be translated and not only those in the 10.130.36.0 network. For this reason, you should specify the network mask when configuring an IP address that is not classful. You must also specify the mask to set other options, such as **outside**.

max_conns	Specifies the maximum number of simultaneous TCP and UDP connections for the entire subnet. The default is 0, which means unlimited connections. (Idle connections are closed after the idle timeout specified by the **timeout conn** command.) **Note** This option does not apply to outside NAT. The firewall only tracks connections from a higher security interface to a lower security interface. If you set *max_conns* as well as the **outside** option, the *max_conns* option is ignored.
nat_id	Specifies an integer for the NAT ID. For regular NAT, this integer is between 1 and 2147483647. For policy NAT (**nat** *id* **access-list**), this integer is between 1 and 65535. Identity NAT (**nat 0**) and NAT exemption (**nat 0 access-list**) use the NAT ID of **0**. See the "nat 0 (Identity NAT)" section on page 7-18 and the "nat 0 access-list (NAT Exemption)" section on page 7-18 for more information about NAT identity and exemption.
norandomseq	Disables TCP Initial Sequence Number (ISN) randomization protection. Only use this option if another inline firewall is also randomizing sequence numbers and the result is scrambling the data. Without this protection, inside hosts with weak self-ISN protection become more vulnerable to TCP connection hijacking. Unless you enable the **norandomseq** option, RCP connections may show a noticable delay with TCP/IP stacks that quickly reuse TCP connections before the timewait state has expired (such as IBM AIX or HP-UX) . **Note** This option does not apply to outside NAT. The firewall only randomizes the ISN that is generated by the host/server on the higher security interface. If you set **norandomseq** as well as the **outside** option, the **norandomseq** option is ignored.
outside	If this interface is on a lower security level than the interface you identify by the matching **global** statement, then you must enter **outside**. This feature is called outside NAT or bidirectional NAT. **Note** Starting with PIX Firewall 6.3.2, source translation is performed before destination translation. For this reason, if the source NAT policy allows the connection, the xlate will be created, even if the traffic is denied by the destination policy.

Command Modes

Configuration mode.

Usage Guidelines

Network Address Translation (NAT) substitutes the local address of a packet with a global address that is routable on the destination network.

When hosts on a higher security interface (inside) access hosts on a lower security interface (outside), you must configure NAT on the inside hosts *or* specifically configure the inside interface to bypass NAT.

An inside host can communicate with the untranslated local address of the outside host without any special configuration on the outside interface. However, you can also optionally perform NAT on the outside network.

The **nat** command identifies the local addresses for translation using dynamic NAT or port address translation (PAT). The **global** command identifies the global addresses used for translation on a given destination interface. Each **nat** statement matches a **global** statement by comparing the NAT ID on each statement. If you bypass NAT using identity NAT or NAT exemption, then no **global** command is required. See the "nat 0 (Identity NAT)" section on page 7-18 and the "nat 0 access-list (NAT Exemption)" section on page 7-18 for more information on bypassing NAT.

After changing or removing a **nat** command statement, use the **clear xlate** command.

You can use the **no nat** command to remove a **nat** command statement.

Note The firewall does not support NAT for a Call Manager (CM) inside the firewall with IP phones outside the firewall (that need to register with it). This is because when the IP phone needs to register with the CM it does so through TFTP, but the firewall does not NAT TFTP messages.

The PIX Firewall does not support outside NAT for non-H.323 multimedia applications or between overlapping network addresses.

Dynamic NAT and PAT

Dynamic NAT translates a group of local addresses to a pool of global addresses that are routable on the destination network. The global pool can include fewer addresses than the local group. When a local host accesses the destination network, the FWSM assigns it an IP address from the global pool. Because the translation is only in place for the duration of the connection, a given user does not keep the same IP address between connections. Users on the destination network, therefore, cannot reliably initiate a connection to a host that uses dynamic NAT (even if the connection is allowed by an access list). Not only can you not predict the IP address of the host, but the host does not have a global address unless the host is the initiator. See the **static** command for reliable access to hosts.

PAT translates a group of local addresses to a single global IP address combined with a unique source port (above 1024). When a local host accesses the destination network, the FWSM assigns it the global IP address and then a unique port number. Each host receives the same IP address, but because the source port numbers are unique, the responding traffic, which includes the IP address and port number as the destination, can be assigned to the correct host. Because there are over 64,000 ports available, you are unlikely to run out of addresses, which can happen with dynamic NAT.

Like dynamic NAT, the translation is only in place for the duration of the connection, so a given user does not keep the same port number between connections.

PAT allows you to use a single global address, thus conserving routable addresses. You can even use the destination interface IP address as the PAT address. PAT does not work with multimedia applications that have an inbound data stream different from the outgoing control path.

Dynamic NAT has these disadvantages:

- If the global pool has fewer addresses than the local group, you could run out of addresses if the traffic is more than expected.

  Use PAT if this event occurs often.

- You have to use a large number of routable addresses in the global pool; if the destination network requires registered addresses, such as the Internet, you might encounter a shortage of usable addresses.

The advantage of dynamic NAT is that some protocols cannot use PAT, which does not work with applications that have an inbound data stream on one port and the outgoing control path on another, such as multimedia applications.

nat Vs. static Commands

The rule of thumb is that for access from a higher security level interface to a lower security level interface, use the **nat** command. From lower security level interface to a higher security level interface, use the **static** command.

Table 7-1 helps you decide when to use the **nat** or **static** commands for access between the various interfaces in the PIX Firewall. For this table, assume that the security levels are 40 for dmz1 and 60 for dmz2.

Table 7-1 Interface Access Commands by Interface

From This Interface	To This Interface	Use This Command
inside	outside	**nat**
inside	dmz1	**nat**
inside	dmz2	**nat**
dmz1	outside	**nat**
dmz1	dmz2	**static**
dmz1	inside	**static**
dmz2	outside	**nat**
dmz2	dmz1	**nat**
dmz2	inside	**static**
outside	dmz1	**static**
outside	dmz2	**static**
outside	inside	**static**

nat 0 (Identity NAT)

The **nat 0** command enables identity NAT. Use this command to bypass NAT and allow the local addresses to be used unchanged. Adaptive Security remains in effect with the **nat 0** command. Both the **nat 0** command and the **nat 0 access-list** command (NAT exemption) may be configured concurrently in PIX Firewall software Version 5.3 and higher.

It is important to understand the difference between identity NAT and NAT exemption. With identity NAT, you can accept the inbound traffic only when the traffic is initiated from the inside and after the xlate is created. NAT exemption allows traffic whenever it matches the referenced ACL, regardless of whether or not there is already an xlate. Identity NAT allows you to set additional NAT parameters, such as **norandomseq**. NAT exemption allows only the **outside** option.

The **nat 0 10.2.3.0** command means let those IP addresses in the 10.2.3.0 net appear on the outside without translation. All other hosts are translated depending on how their **nat** or **static** command statements appear in the configuration.

nat 0 access-list (NAT Exemption)

The **nat 0 access-list** command disables NAT, specifically proxy ARPing, for the IP addresses specified by the ACL referenced by *acl_id*. (The *acl_id* is the name you use to identify the **access-list** command statement.) This feature is known as NAT exemption. NAT exemption is not backward compatible with PIX Firewall software Version 5.2 or earlier versions.

This feature is useful in a Virtual Private Network (VPN) configuration where traffic between private networks should be exempted from NAT.

While NAT exemption lets you exempt traffic that is matched by the **access-list** command statement from NAT services, Adaptive Security remains in effect. The extent to which the inside hosts are accessible from the outside depends on the **access-list** command statements that permit inbound access; NAT exemption allows both inbound and outbound traffic no matter which side initiates, as long as it is permitted by the referenced ACL.

Unlike policy NAT, the PIX Firewall ignores any port setting in your ACL command statement and so NAT exemption cannot be used to permit or deny traffic on a per-port basis.

nat outside (Outside NAT)

The **nat outside** option lets you enable or disable outside NAT, which translates the source address of a connection coming from a lower security interface to higher interface. This feature is also called bidirectional NAT.

If you enable outside dynamic NAT on an interface, then you must configure explicit NAT policy for all hosts on the interface that need to initiate connections to inside networks. If you want to translate some hosts, but not others, then use identity NAT or NAT exemption (**nat 0** or **nat 0 access-list**) to disable address translation for these additional hosts.

The **norandomseq** and **emb_limit** options are not supported with outside NAT.

Note Enabling outside PAT can make the firewall more susceptible to flood DoS attack. To mitigate this, we recommend that the address range selected with the **nat** *nat_id local_ip mask* **outside** command be as restrictive as possible. In addition, the connection limit should be set to a value that takes into consideration the memory capacity of the firewall. In general, a PAT session is composed of a PAT xlate and a UDP or TCP connection. A PAT xlate consumes about 120 bytes and a TCP or UDP connection consumes about 250 bytes.

nat *nat_id* access-list (Policy NAT)

When you use an access list with the **nat** command for any NAT ID other than 0, then you enable policy NAT.

Policy NAT lets you identify local traffic for address translation by specifying the source and destination addresses (or ports) in an access list. Regular NAT uses source addresses/ports only, whereas policy NAT uses both source and destination addresses/ports.

Note All types of NAT support policy NAT except for NAT exemption (**nat 0 access-list**). NAT exemption uses an ACL to identify the local addresses, but differs from policy NAT in that the ports are not considered.

With policy NAT, you can create multiple NAT or static statements that identify the same local address as long as the source/port and destination/port combination is unique for each statement. You can then match different global addresses to each source/port and destination/port pair.

Order of NAT Commands Used to Match Local Addresses

The firewall matches local traffic to NAT commands in the following order:

1. **nat 0 access-list** (NAT exemption)—In order, until the first match. For example, you could have overlapping local/destination addresses in multiple **nat** commands, but only the first command is matched.

2. **static** (static NAT)—In order, until the first match. Because you cannot use the same local address in static NAT or static PAT commands, the order of **static** commands does not matter. Similarly, for static policy NAT, you cannot use the same local/destination address and port across multiple statements.
3. **static** {**tcp** | **udp**} (static PAT)—In order, until the first match. Because you cannot use the same local address in static NAT or static PAT commands, the order of **static** commands does not matter. Similarly, for static policy NAT, you cannot use the same local/destination address and port across multiple statements.
4. **nat** *nat_id* **access-list** (policy NAT)—In order, until the first match. For example, you could have overlapping local/destination ports and addresses in multiple **nat** commands, but only the first command is matched.
5. **nat** (regular NAT)—Best match. The order of the NAT commands does not matter. The **nat** statement that best matches the local traffic is used. For example, you can create a general statement to translate all addresses (0.0.0.0) on an interface. If you also create a statement to translate only 10.1.1.1, when 10.1.1.1 makes a connection, the specific statement for 10.1.1.1 is used because it matches the local traffic best.

If you configure multiple **global** statements on the same NAT ID, the **global** statements are used in this order:

1. No **global** if using **nat 0** (identity NAT).
2. Dynamic NAT **global**.
3. PAT **global**.

Examples

The **nat 0** (identity NAT) command allows traffic to be initiated from the local host only.

If you want the addresses to be visible from the outside network, use NAT exemption, or use the **static** command as follows:

```
nat (inside) 0 209.165.201.0 255.255.255.224
static (inside, outside) 209.165.201.0 209.165.201.0 netmask 255.255.255.224
access-list acl_out permit host 10.0.0.1 209.165.201.0 255.255.255.224 eq ftp
access-group acl_out in interface outside

nat (inside) 0 209.165.202.128 255.255.255.224
static (inside, outside) 209.165.202.128 209.165.202.128 netmask 255.255.255.255
access-list acl_out permit tcp host 10.0.0.1 209.165.202.128 255.255.255.224 eq ftp
access-group acl_out in interface outside
```

The following example shows use of the **nat 0 access-list** command (NAT exemption) to permit internal host 10.1.1.15, which is accessible through the inside interface, to bypass NAT when connecting to outside host 10.2.1.3.

```
access-list no-nat permit ip host 10.1.1.15 host 10.2.1.3
nat (inside) 0 access-list no-nat
```

The following commands use NAT exemption on a PIX Firewall with three interfaces:

```
access-list all-ip-packet permit ip 0 0 0 0
nat (dmz) 0 access-list all-ip-packet
nat (inside) 0 access-list all-ip-packet
```

Given outbound traffic and the following example, for the **nat** command statements with a *nat_id* of **1**, any of the hosts on the 10.1.1.0 network are translated to the range of 209.165.201.25-209.165.201.27. After all three addresses have been used, the translation rule starts using 209.165.201.30 as the PAT address. For the **nat** command statements with a *nat_id* of **3**, all of the hosts on the 10.1.3.0 network are translated to the outside IP address of the FWSM using PAT.

```
nat (inside) 1 10.1.1.0 255.255.255.0
global (outside) 1 209.165.201.25-209.165.201.27 netmask 255.255.255.224
global (outside) 1 209.165.201.30

nat (inside) 3 10.1.3.0 255.255.255.0
global (outside) 3 209.165.201.30
```

The following example specifies with **nat** command statements that all the hosts on the 10.0.0.0 and 10.3.3.0 inside networks can start outbound connections. The **global** command statements create unique pools of global addresses for those hosts that cannot overlap.

```
nat (inside) 1 10.0.0.0 255.0.0.0
global (outside) 1 209.165.201.24-209.165.201.27 netmask 255.255.255.224
global (outside) 1 209.165.201.30

nat (inside) 3 10.3.3.0 255.255.255.0
global (outside) 3 209.165.201.10-209.165.201.23 netmask 255.255.255.224
```

The following policy NAT example shows a host on the 10.1.2.0/24 network accessing two different servers. When the host accesses the server at 209.165.201.11, the local address is translated to 209.165.202.129. When the host accesses the server at 209.165.200.225, the local address is translated to 209.165.202.130.

```
access-list NET1 permit ip 10.1.2.0 255.255.255.0 209.165.201.0 255.255.255.224
access-list NET2 permit ip 10.1.2.0 255.255.255.0 209.165.200.224 255.255.255.224
nat (inside) 1 access-list NET1
global (outside) 1 209.165.202.129 255.255.255.255
nat (inside) 2 access-list NET2
global (outside) 2 209.165.202.130 255.255.255.255
```

The following policy NAT example shows the use of source and destination ports. The host on the 10.1.2.0/24 network accesses a single host for both web services and Telnet services. When the host accesses the server for web services, the local address is translated to 209.165.202.129. When the host accesses the same server for Telnet services, the local address is translated to 209.165.202.130.

```
access-list WEB permit tcp 10.1.2.0 255.255.255.0 209.165.201.11 255.255.255.255 eq 80
access-list TELNET permit tcp 10.1.2.0 255.255.255.0 209.165.201.11 255.255.255.255 eq 23
nat (inside) 1 access-list WEB
global (outside) 1 209.165.202.129 255.255.255.255
nat (inside) 2 access-list TELNET
global (outside) 2 209.165.202.130 255.255.255.255
```

Related Commands

access-list	Configures access control.
global	Configures global address pools, or designates a PAT (Port Address Translation) address.
interface	Sets network interface parameters and configures VLANs.
nameif	Assigns a name to an interface.
static	Configures a one-to-one address translation rule.

ntp

Synchronizes the PIX Firewall with a network time server using the Network Time Protocol (NTP).

[**no**] **ntp authenticate**

[**no**] **ntp authentication-key** *number* **md5** *value*

ntp server *ip_address* [**key** *number*] **source** *if_name* [**prefer**]

no ntp server *ip_address*

[**no**] **ntp trusted-key** *number*

clear ntp

show ntp

show ntp associations [**detail**]

show ntp status

Syntax Description

associations	The network time server associations.
authenticate	Enables NTP authentication. If enabled, the PIX Firewall requires authentication before synchronizing with an NTP server.
authentication-key	Defines the authentication keys for use with other NTP commands.
detail	Provides additional detail on the network time servers.
if_name	Specifies the interface to use to send packets to the network time server.
ip_address	The IP address of the network time server with which to synchronize.
key	Specifies the authentication key.
md5	The encryption algorithm.
number	The authentication key number (1 to 4294967295).
prefer	Designates the network time server specified as the preferred server with which to synchronize time.
server	The network time server.
source	Specifies the network time source.
status	Displays NTP clock information.
trusted-key	Specifies the trusted key against which to authenticate.
value	The key value, an arbitrary string of up to 32 characters. The key value is displayed as "**********" when the configuration is viewed by the **write terminal** or **show tech-support** commands.

Command Modes

Configuration mode.

Usage Guidelines

The **ntp** command synchronizes the PIX Firewall with the network time server that is specified and authenticates according to the authentication options that are set.

The authentication keys for the **ntp** commands are defined in the **ntp authentication-key** command. If authentication is used, the PIX Firewall and NTP server must be configured with the same key.

If authentication is enabled, use the **ntp trusted-key** command to define one or more key numbers that the NTP server needs to provide in its NTP packets for the PIX Firewall to accept synchronization with the NTP server.

The PIX Firewall listens for NTP packets (port 123) only on interfaces that have an NTP server configured through the **ntp server** command. NTP packets that are not responses from a request by the PIX Firewall are dropped.

The **ntp authenticate** command enables NTP authentication.

The **clear ntp** command removes the NTP configuration, including disabling authentication and removing all authentication keys and NTP server designations.

show ntp commands

To view information about the NTP configuration and status, use the **show ntp, show ntp associations** [**detail**], or **show ntp status** commands.

The **show ntp** command displays the current NTP configuration.

The **show ntp associations** [**detail**] command displays the configured network time server associations.

The **show ntp status** command displays the NTP clock information.

The following is sample output from the **show ntp associations** command:

```
pixfirewall> show ntp associations
     address         ref clock     st  when  poll  reach  delay  offset    disp
 ~172.31.32.2      172.31.32.1       5    29  1024  377     4.2   -8.59     1.6
+~192.168.13.33    192.168.1.111     3    69   128  377     4.1    3.48     2.3
*~192.168.13.57    192.168.1.111     3    32   128  377     7.9   11.18     3.6
* master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

Table 7-2 describes the values in the **show ntp associations** command output:

Table 7-2 Output Description from ntp association Command

Output	Description
*	Synchronized to this peer
#	Almost synchronized to this peer
+	Peer selected for possible synchronization
-	Peer is a candidate for selection
~	Peer is statically configured
address	Address of peer.
ref clock	Address of reference clock of peer.
st	Stratum of peer.
when	Time since last NTP packet was received from peer.
poll	Polling interval (in seconds).
reach	Peer reachability (bit string, in octal).
delay	Round-trip delay to peer (in milliseconds).

Table 7-2 Output Description from ntp association Command

Output	Description
offset	Relative time of peer clock to local clock (in milliseconds).
disp	Dispersion.

The following is sample output from the **show ntp association detail** command:

```
pixfirewall(config)# show ntp associations detail
172.23.56.249 configured, our_master, sane, valid, stratum 4
ref ID 172.23.56.225, time c0212639.2ecfc9e0 (20:19:05.182 UTC Fri Feb 22
2002)
our mode client, peer mode server, our poll intvl 128, peer poll intvl 128
root delay 38.04 msec, root disp 9.55, reach 177, sync dist 156.021
delay 4.47 msec, offset -0.2403 msec, dispersion 125.21
precision 2**19, version 3
org time c02128a9.731f127b (20:29:29.449 UTC Fri Feb 22 2002)
rcv time c02128a9.73c1954b (20:29:29.452 UTC Fri Feb 22 2002)
xmt time c02128a9.6b3f729e (20:29:29.418 UTC Fri Feb 22 2002)
filtdelay =     4.47    4.58    4.97    5.63    4.79    5.52    5.87
0.00
filtoffset =   -0.24   -0.36   -0.37    0.30   -0.17    0.57   -0.74
0.00
filterror =     0.02    0.99    1.71    2.69    3.66    4.64    5.62
16000.0
```

Table 7-3 describes the values in the **show ntp association detail** command output:

Table 7-3 Output Description from ntp association detail Command

Output	Description
configured	Peer was statically configured.
dynamic	Peer was dynamically discovered.
our_master	Local machine is synchronized to this peer.
selected	Peer is selected for possible synchronization.
candidate	Peer is a candidate for selection.
sane	Peer passes basic sanity checks.
insane	Peer fails basic sanity checks.
valid	Peer time is believed to be valid.
invalid	Peer time is believed to be invalid.
leap_add	Peer is signalling that a leap second will be added.
leap-sub	Peer is signalling that a leap second will be subtracted.
unsynced	Peer is not synchronized to any other machine.
ref ID	Address of machine peer is synchronized to.
time	Last time stamp peer received from its master.
our mode	Our mode relative to peer (active/passive/client/server/bdcast/bdcast client).
peer mode	Peer's mode relative to us.
our poll intvl	Our poll interval to peer.

Table 7-3 Output Description from ntp association detail Command (continued)

Output	Description
peer poll intvl	Peer's poll interval to us.
root delay	Delay along path to root (ultimate stratum 1 time source).
root disp	Dispersion of path to root.
reach	Peer reachability (bit string in octal).
sync dist	Peer synchronization distance.
delay	Round-trip delay to peer.
offset	Offset of peer clock relative to our clock.
dispersion	Dispersion of peer clock.
precision	Precision of peer clock in hertz.
version	NTP version number that peer is using.
org time	Originate time stamp.
rcv time	Receive time stamp.
xmt time	Transmit time stamp.
filtdelay	Round-trip delay (in milliseconds) of each sample.
filtoffset	Clock offset (in milliseconds) of each sample.
filterror	Approximate error of each sample.

The following is sample output from the **show ntp status** command:

```
pixfirewall(config)# show ntp status
Clock is synchronized, stratum 5, reference is 172.23.56.249
nominal freq is 99.9984 Hz, actual freq is 100.0266 Hz, precision is 2**6
reference time is c02128a9.73c1954b (20:29:29.452 UTC Fri Feb 22 2002)
clock offset is -0.2403 msec, root delay is 42.51 msec
root dispersion is 135.01 msec, peer dispersion is 125.21 msec
```

Table 7-4 describes the values in the **show ntp status** command output:

Table 7-4 Output Description from ntp status Command

Output	Description
synchronized	System is synchronized to an NTP peer.
unsynchronized	System is not synchronized to any NTP peer.
stratum	NTP stratum of this system.
reference	Address of peer to which the system is synchronized.
nominal freq	Nominal frequency of system hardware clock.
actual freq	Measured frequency of system hardware clock.
precision	Precision of the clock of this system (in hertz).
reference time	Reference time stamp.
clock offset	Offset of the system clock to synchronized peer.
root delay	Total delay along path to root clock.

Table 7-4 Output Description from ntp status Command (continued)

Output	Description
root dispersion	Dispersion of root path.
peer dispersion	Dispersion of synchronized peer.

Examples

The following is sample output from the **show ntp** command:

```
pixfirewall(config)# show ntp
ntp authentication-key 1234 md5 ********
ntp authenticate
ntp trusted-key 1234
ntp server 10.10.1.2 key 1234 source inside prefer
pixfirewall(config)#
```

The following is sample output from the **show ntp associations** command:

```
pixfirewall(config)# show ntp associations
  address           ref clock         st  when  poll reach   delay  offset  disp
 *~172.23.56.249    172.23.56.225     4   113   128  177     4.5   -0.24   125.2
 * master (synced), # master (unsynced), + selected, - candidate, ~ configured
```

The following is sample output from the **show ntp associations detail** command:

```
pixfirewall(config)# show ntp associations detail
172.23.56.249 configured, our_master, sane, valid, stratum 4
ref ID 172.23.56.225, time c0212639.2ecfc9e0 (20:19:05.182 UTC Fri Feb 22 2002)
our mode client, peer mode server, our poll intvl 128, peer poll intvl 128
root delay 38.04 msec, root disp 9.55, reach 177, sync dist 156.021
delay 4.47 msec, offset -0.2403 msec, dispersion 125.21
precision 2**19, version 3
org time c02128a9.731f127b (20:29:29.449 UTC Fri Feb 22 2002)
rcv time c02128a9.73c1954b (20:29:29.452 UTC Fri Feb 22 2002)
xmt time c02128a9.6b3f729e (20:29:29.418 UTC Fri Feb 22 2002)
filtdelay =     4.47    4.58    4.97    5.63    4.79    5.52    5.87    0.00
filtoffset =   -0.24   -0.36   -0.37    0.30   -0.17    0.57   -0.74    0.00
filterror =     0.02    0.99    1.71    2.69    3.66    4.64    5.62    16000.0
```

The following is sample output from the **show ntp status** command:

```
pixfirewall(config)# show ntp status
Clock is synchronized, stratum 5, reference is 172.23.56.249
nominal freq is 99.9984 Hz, actual freq is 100.0266 Hz, precision is 2**6
reference time is c02128a9.73c1954b (20:29:29.452 UTC Fri Feb 22 2002)
clock offset is -0.2403 msec, root delay is 42.51 msec
root dispersion is 135.01 msec, peer dispersion is 125.21 msec
```

Related Commands

clock	Sets the date and time of firewall.

object-group

Defines object groups that you can use to optimize your configuration. Objects such as hosts, protocols, or services can be grouped, and then you can issue a single command using the group name to apply to every item in the group.

[**no**] **object-group icmp-type** *grp_id*

ICMP type group subcommands:
description *description_text*
icmp-object *icmp_type*
group-object *grp_id*

[**no**] **object-group network** *grp_id*

network group subcommands:
description *description_text*
network-object host *host_addr*
network-object *host_addr mask*
group-object *grp_id*

[**no**] **object-group protocol** *grp_id*

protocol group subcommands:
description *description_text*
protocol-object *protocol*
group-object *grp_id*

[**no**] **object-group service** *grp_id* {**tcp | udp | tcp-udp**}

service group subcommands:
description *description_text*
port-object range *begin_service end_service*
port-object eq *service*
group-object *grp_id*

clear object-group [*grp_type]*

show object-group [**id** *grp_id* | *grp_type*]

Note Enter **no** in front of a subcommand to remove the configuration within an object group.

Syntax Description.

begin_service	Used with the **range** keyword, the decimal number or name of a TCP or UDP port that is the beginning value for a range of services.
description *description_text*	A subcommand of the **object-group** command that enables users to add a description of up to 200 characters to an object-group. The starting position of the description text is the character right after the whitespace (a blank or a tab) following the **description** keyword.
end_service	Used with the **range** keyword, the decimal number or name of a TCP or UDP port that is the ending value for a range of services.

eq *service*	Specifies the decimal number or name of a TCP or UDP port for a particular service object.
group-object	The **group-object** subcommand is used to add a group of objects that are themselves members of another object group.
grp_id	Required parameter that identifies the object group (one to 64 characters). Can be any combination of letters, digits, and the "_", "-", "." characters.
grp_type	*The type of group, either ICMP type, network, protocol, or service.*
host	Keyword used with the *host_addr* parameter to define a host object.
host_addr	The host IP address or host name (if the host name is already defined using the **name** command).
icmp-object	The **object-group icmp-type** subcommand used to add ICMP objects to an ICMP-type object group.
icmp-type	Defines a group of ICMP types such as echo and echo-reply. After entering the main **object-group icmp-type** command, add ICMP objects to the ICMP type group with the **icmp-object** and the **group-object** subcommand.
icmp_type	The decimal number or name of an ICMP type.
mask	The netmask. Used with *net_addr* to define a subnet object.
net_addr	The network address. Used with *netmask* to define a subnet object.
network	Defines a group of hosts or subnet IP addresses. After entering the main **object-group network** command, add network objects to the network group with the **network-object** and the **group-object** subcommand.
network-object	The **object-group network** subcommand used to add network objects to a network object group.
obj_grp_id	The name of a previously defined object group. For object groups to be grouped together, they must be of the same type. For example, you can group two or more network object groups together, but you cannot group a protocol group and a network group together.
object-group	The main object grouping command. The keyword after it specifies the type of object group that is being defined. After entering this main command with the type indicator keyword, you are in subcommand mode where you explicitly define individual group members using the **object-group** subcommands.
port-object	The **object-group service** subcommand used to add port objects to a service object group.
protocol	Defines a group of protocols such as TCP and UDP. After entering the main **object-group protocol** command, add protocol objects to the protocol group with the **protocol-object** and the **group-object** subcommand.
protocol	The protocol name or number. (For example, UDP is 17 and TCP is 6.)
protocol-object	The **object-group protocol** subcommand used to add protocol objects to a protocol object group.
range	Keyword indicating that the range parameters follow.
service	Defines a group of TCP/UDP port specifications such as "eq smtp" and "range 2000 2010." After entering the main **object-group service** command, add port objects to the service group with the **port-object** and the **group-object** subcommand.
tcp	Specifies that service group is used for TCP.

tcp-udp	Specifies that service group can be used for TCP and UDP.
udp	Specifies that service group is used for UDP.

Command Modes

Configuration mode.

Usage Guidelines

When a group is defined with the **object-group** command and then used in a PIX Firewall command, the command applies to every item in that group. This can significantly reduce your configuration size.

Once an object group is defined, the keyword **object-group** must be used before the group name in all applicable PIX Firewall commands, for example:

show object-group *group_name*

where *group_name* is the name of the group.

The following are two examples of the use of an object group once it is defined:

```
conduit permit tcp object-group group_name any
access-list acl_id permit tcp any object-group group_name
```

Additionally, the **access-list** and **conduit** command parameters can be grouped as follows in Table 7-5.

Table 7-5 Object Groups to Replace Individual Parameters

Instead of using individual parameters...	...use the following object group:
protocol	**object-group** *protocol*
host and subnet	**object-group** *network*
service	**object-group** *service*
icmp_type	**object-group** *icmp_type*

You can group commands hierarchically; an object group can be a member of another object group.

To use object groups, you must do the following:

- The keyword **object-group** must be used before the object group name in all commands.

  For example:

  ```
  access-list acl permit tcp object-group remotes object-group locals object-group
  eng_svc
  ```

  where *remotes* and *locals* are sample object group names.
- The object group must be non-empty.
- An object group cannot be removed or emptied if it is currently being used in a command.

After a main **object-group** command is entered, the command mode changes to its corresponding subcommand mode. The object group is then defined in the subcommand mode. The active mode is indicated in the command prompt format. For example, the prompt in the configuration terminal mode appears as follows:

```
pix_name (config)#
```

where *pix_name* is the name of the PIX Firewall.

However, when the **object-group** command is entered, the prompt appears as follows:

```
pix_name (config-type)#
```

where *pix_name* is the name of the PIX Firewall and *type is the object-group type.*

Use **exit**, **quit**, or any valid config-mode command such as the **access-list** command to close an **object-group** subcommand mode and exit the **object-group** main command.

object groupingUse the **no object-group** command form to remove a group of previously defined **object-group** commands. The **clear object-group** command form can also be used.

The **show object-group** command displays all defined object groups by their *grp_id* when the **show object-group id** *grp_id* command form is entered, and by their group type when the **show object-group** *grp_type* command form is entered. When you enter the **show object-group** command without a parameter, all defined object groups are shown.

When entered without a parameter, the **clear object-group** command removes all defined object groups that are not being used in a command. Using *grp_type* parameter removes all defined object groups that that are not being used in a command for that group type only.

For use in the **object-group icmp-type** command, Table 7-6 lists ICMP type numbers and names:

Table 7-6 object groupingICMP Types

Number	Name of ICMP Type
0	echo-reply
3	unreachable
4	source-quench
5	redirect
6	alternate-address
8	echo
9	router-advertisement
10	router-solicitation
11	time-exceeded
12	parameter-problem
13	timestamp-request
14	timestamp-reply
15	information-request
16	information-reply
17	mask-request
18	mask-reply
31	conversion-error
32	mobile-redirect

Usage Notes

1. You can use all other PIX Firewall commands in subcommand mode, including the **show** and **clear** commands.
2. Subcommands appear indented when displayed or saved by the **show config**, **write**, or **config** commands.

3. Subcommands have the same command privilege level as the main command.
4. When more than one object group is used in an **access-list** or **conduit** command, the elements of all object groups used in the command are cross-concatenated together, starting with the first group's elements concatenated the second group's elements, then the first and second group's elements concatentated together with the third group's elements, and so on.

Examples

The following example shows how to use the **object-group icmp-type** subcommand mode to create a new icmp-type object group:

```
pixfirewall(config)# object-group icmp-type icmp-allowed
    pixfirewall(config-icmp-type)#icmp-object echo
    pixfirewall(config-icmp-type)#icmp-object time-exceeded
    pixfirewall(config-icmp-type)#exit
```

The following example shows how to use the **object-group network** subcommand to create a new network object group:

```
pixfirewall(config)# object-group network sjc_eng_ftp_servers
    pixfirewall(config-network)#network-object host sjc.eng.ftp.servcers
    pixfirewall(config-network)#network-object host 172.23.56.194
    pixfirewall(config-network)#network-object 192.1.1.0 255.255.255.224
    pixfirewall(config-network)#exit
```

The following example shows how to use the **object-group network** subcommand to create a new network object group and map it to a existing object-group:

```
pixfirewall(config)# object-group network sjc_ftp_servers
    pixfirewall(config-network)#network-object host sjc.ftp.servers
    pixfirewall(configpixfirewall(config-network)#network-object host 172.23.56.195
    pixfirewall(config-network)#network-object 193.1.1.0 255.255.255.224
    pixfirewall(config-network)#group-object sjc_eng_ftp_servers
    pixfirewall(config-network)#exit
```

The following example shows how to use the **object-group protocol** subcommand mode to create a new protocol object group:

```
pixfirewall(config)# object-group protocol proto_grp_1
    pixfirewall(config-protocol)#protocol-object udp
    pixfirewall(config-protocol)#protocol-object ipsec
    pixfirewall(config-protocol)#exit

pixfirewall(config)# object-group protocol proto_grp_2
    pixfirewall(config-protocol)#protocol-object tcp
    pixfirewall(config-protocol)#group-object proto_grp_1
    pixfirewall(config-protocol)#exit
```

The following example shows how to use the **object-group service** subcommand mode to create a new port (service) object group:

```
pixfirewall(config)# object-group service eng_service tcp
    pixfirewall(config-service)#group-object eng_www_service
    pixfirewall(config-service)#port-object eq ftp
    pixfirewall(config-service)#port-object range 2000 2005
    pixfirewall(config-service)#exit
```

The following example shows how to add and remove a text description to an object group:

```
pixfirewall(config)# object-group protocol protos1
    pixfirewall(config-protocol)# description This group of protocols is for our internal
    network
```

```
    pixfirewall(config-protocol)# show object-group id protos1
    object-group protocol protos1
      description: This group of protocols is for our internal network

    pixdocipsec1(config-protocol)# no description
    pixdocipsec1(config-protocol)# show object-group id protos1
    object-group protocol protos1
```

The following example shows how to use the **object groupinggroup-object** subcommand mode to create a new object group that consists of previously defined objects:

```
pixfirewall(config)# object-group network host_grp_1
    pixfirewall(config-network)# network-object host 192.168.1.1
    pixfirewall(config-network)# network-object host 192.168.1.2
    pixfirewall(config-network)# exit

pixfirewall(config)# object-group network host_grp_2
    pixfirewall(config-network)# network-object host 172.23.56.1
    pixfirewall(config-network)# network-object host 172.23.56.2
    pixfirewall(config-network)# exit

pixfirewall(config)# object-group network all_hosts
    pixfirewall(config-network)# group-object host_grp_1
    pixfirewall(config-network)# group-object host_grp_2
    pixfirewall(config-network)# exit

pixfirewall(config)# access-list grp_1 permit tcp object-group host_grp_1 any eq ftp
pixfirewall(config)# access-list grp_2 permit tcp object-group host_grp_2 any eq smtp
pixfirewall(config)# access-list all permit tcp object-group all_hosts any eq www
```

As shown in this example, without the **group-object** command the *all_hosts* group has to be defined to include all the IP addresses that have already defined in *host_grp_1* and *host_grp_2*, but with the **group-object** command, the duplicated definitions of the hosts are eliminated.

The following example illustrates how use object groups to simplify access list configuration:

```
object-group network remote
    network-object host kqk.suu.dri.ixx
    network-object host kqk.suu.py1.gnl

object-group network locals
    network-object host 172.23.56.10
    network-object host 172.23.56.20
    network-object host 172.23.56.194
    network-object host 172.23.56.195

object-group service eng_svc ftp
    port-object eq www
    port-object eq smtp
    port-object range 25000 25100
```

This grouping then enables the access list to be configured in one line instead of 24 lines, which would be needed if no grouping is used. Instead, with the grouping, the access list configuration is as follows:

```
access-list acl permit tcp object-group remote object-group locals object-group eng_svc
```

Note The **show config** and **write** commands display the access list as configured with the object group names. However, the **show access-list** command displays the access list entries expanded out into individual statements without their object groupings.

outbound/apply

Create an access list for controlling Internet use.

[no] apply [(*if_name*)] *list_ID* **outgoing_src | outgoing_dest**

clear apply

[no] outbound *list_ID* **permit | deny** *ip_address* [*netmask* [*port*[**-***port*]] [*protocol*]

[no] outbound *list_ID* **except** *ip_address* [*netmask* [*port*[**-***port*]] [*protocol*]

clear outbound

show apply [(*if_name*)] [*list_ID* **outgoing_src | outgoing_dest**]

show outbound

Syntax Description

apply	Specifies whether the access control list applies to inside users' ability to start outbound connections with **apply** command's **outgoing_src** option, or whether the access list applies to inside users' ability to access servers on the outside network with the **apply** command's **outgoing_dest** option.
clear apply	Removes all the **apply** command statements from the configuration.
clear outbound	Removes all **outbound** command statements from the configuration.
deny	Deny the access list access to the specified IP address and port.
except	Create an exception to a previous **outbound** command. An **except** command statement applies to **permit** or **deny** command statements only with the same access list ID. When used with **apply outgoing_src**, the IP address of an **except** command statement applies to the destination address. When used with **apply outgoing_dest**, the IP address of an **except** command statement applies to the source address. See "Outbound List Rules" for more information.
if_name	The network interface originating the connection.
ip_address	The IP address for this access list entry. Do not specify a range of addresses. The 0.0.0.0 *ip_address* can be abbreviated as 0.
list_ID	A tag number for the access list. The access list number you use must be the same for the **apply** and **outbound** commands. This value must be a positive number from 1 to 1599. This number can be the same as what you use with the **nat** and **global** commands. This number is just an arbitrary number that groups **outbound** command statements to an **apply** command statement. *List_IDs* are processed sequentially in descending order. For more information, see "Outbound List Rules."
netmask	The network mask for comparing with the IP address; 255.255.255.0 causes the access list to apply to an entire Class C address. 0.0.0.0 indicates all access. The 0.0.0.0 *netmask* can be abbreviated as 0.
no outbound	Removes a single **outbound** command statement from the configuration.

no apply	Removes a single **apply** command statement from the configuration.
outbound	The **outbound** command, in conjunction with the **apply** command, uses access lists to control a filtering function on outgoing packets from the PIX Firewall. The filters can be based on the source IP address, the destination IP address, and the destination port/protocol as specified by the rules. The use of an **outbound** command requires use of the **apply** command. The **apply** command lets you specify whether the access control list applies to inside users' ability to start outbound connections with the **apply** command's **outgoing_src** option, or whether the access list applies to inside users' ability to access servers on the outside network with the **apply** command's **outgoing_dest** option. For more information, see "Outbound List Rules" and the **access-list** command. The **outbound** command has been superseded by the **access-list** command.
outgoing_dest	Deny or permit access to an external IP address using the service(s) specified in the **outbound** command.
outgoing_src	Deny or permit an internal IP address the ability to start outbound connections using the service(s) specified in the **outbound** command.
permit	Allow the access list to access the specified IP address and port.
port	A port or range of ports that the access list is permitted or denied access to. See the "Ports" section in Chapter 2, "Using PIX Firewall Commands" for a list of valid port literal names.
protocol	Limit outbound access to **udp**, **tcp**, or **icmp** protocols. If a protocol is not specified, the default is **tcp**.

Command Modes

Configuration mode.

Usage Guidelines

The **outbound** command creates an access list that lets you specify the following:

- Whether inside users can create outbound connections
- Whether inside users can access specific outside servers
- What services inside users can use for outbound connections and for accessing outside servers
- Whether outbound connections can execute Java applets on the inside network

Outbound lists are filters on outgoing packets from the PIX Firewall. The filter can be based on the source IP address, the destination IP address, and the destination port/protocol as specified by the rules. The use of an **outbound** command requires use of the **apply** command. The **apply** command enables you to specify whether the access control list applies to inside users' ability to start outbound connections with **apply** command's **outgoing_src** option, or whether the access list applies to inside users' ability to access servers on the outside network with the **apply** command's **outgoing_dest** option.

Note The **outbound** command has been superseded by the **access-list** command. We recommend that you migrate your **outbound** command statements to **access-list** command statements to maintain future compatibility.

The **java** option has been replaced by the **filter java** command.

After adding, removing, or changing **outbound** command statements, use the **clear xlate** command.

Use the **no outbound** command to remove a single **outbound** command statement from the configuration. Use the **clear outbound** command to remove all **outbound** command statements from the configuration. The **show outbound** command displays the **outbound** command statements in the configuration.

Use the **no apply** command to remove a single **apply** command statement from the configuration. Use the **clear apply** command statement to remove all the **apply** command statements from the configuration. The **show apply** command displays the **apply** command statements in the configuration.

Outbound List Rules

Rules, written as **outbound** *list_ID* command statements are global to the PIX Firewall; they are activated by **apply** *list_ID outgoing_src* | *outgoing_dest* command statements. When applied to *outgoing_src*, the source IP address, the destination port, and protocol are filtered. When applied to *outgoing_dest*, the destination IP address, port, and protocol are filtered.

The *outgoing_src* option and *outgoing_dest* outbound lists are filtered independently. If any one of the filters contain the **deny** option, the outbound packet is denied. When multiple rules are used to filter the same packet, the best matched rule takes effect. The best match is based on the IP address mask and the port range check. More strict IP address masks and smaller port ranges are considered a better match. If there is a tie, a **permit** option overrides a **deny** option.

Rules are grouped by a *list_ID*. Within each *list_ID*, **except** rules (that is, **outbound** *n* **except** …) can be set. The **except** option reverses the best matched rule of **deny** or **permit**. In addition, PIX Firewall filters the specified IP address and mask in the rule for the destination IP address of the outbound packet if the list is applied to the *outbound_src*. Alternatively, PIX Firewall filters the source IP address if the list is applied to the *outgoing_dest*. Furthermore, the **except** rules only apply to rules with the same *list_ID*. A single **except** rule within a *list_ID* without another **permit** or **deny** rule has no effect. If multiple **except** rules are set, the best match is checked for which **except** to apply.

The **outbound** command rules are now sorted by the best match checking. Use the **show outbound** command to see how the best match is judged by the PIX Firewall.

Usage Notes

1. If **outbound** commands are not specified, the default behavior is to permit all outbound traffic and services from inside hosts.
2. After adding, changing, or removing an **outbound** and **apply** command statement group, use the **clear xlate** command to make the IP addresses available in the translation table.
3. The **outbound** commands are processed linearly within a *list_ID*. In addition, *list_ID*s are processed sequentially in descending order. For example, the first command statement you specify in an **outbound** list is processed first, then the next **outbound** command statement in that list, and so on. Similarly, *list_ID* 10 is processed before *list_ID* 20, and so on.
4. When using **outbound** commands, it is often helpful to deny or permit access to the many before you deny or permit access to the specific. Start with an interface-wide specification such as the following command that denies all hosts from starting connections.

```
outbound 1 deny 0 0 0
apply (inside) 1 outgoing_src
```

Then add command statements that permit or deny hosts access to specific ports.

For example:

```
outbound 1 deny 0 0 0
outbound 1 permit 10.1.1.1 255.255.255.255 23 tcp
outbound 1 permit 10.1.1.1 255.255.255.255 80 tcp
apply (inside) 1 outgoing_src
```

You could state this same example as follows with the **except** option:

```
outbound 1 deny 0 0 0
outbound 1 except 209.165.201.11 255.255.255.255 23 tcp
outbound 1 except 209.165.201.11 255.255.255.255 80 tcp
apply (inside) 1 outgoing_src
```

In the preceding **outbound except** command statement, IP address 209.165.201.11 is the destination IP address, not the source address. This means that everyone is denied outbound access, except those users going to 209.165.201.11 via Telnet (port 23) or HTTP (port 80).

5. If you permit access to port 80 (**http**), this also permits Java applets to be downloaded. You must have a specific **deny** command statement to block Java applets.
6. The maximum number of **outbound** list entries in a configuration is 1599.
7. Outbound lists have no effect on **access-list** command statement groups.
8. The use of the **access-group** command statement overrides the **conduit** and **outbound** command statements for the specified interface name.

Examples

In the following example, the first **outbound** group sets inside hosts so that they can only see and Telnet to perimeter hosts, and do DNS lookups. The perimeter network address is 209.165.201.0 and the network mask is 255.255.255.224.

```
outbound 9 deny 0.0.0.0 0.0.0.0 0 0
outbound 9 except 209.165.201.0 255.255.255.224 23 tcp
outbound 9 except 0.0.0.0 0.0.0.0 53 udp
```

The next **outbound** group lets hosts 10.1.1.11 and 10.1.1.12 go anywhere:

```
outbound 11 deny 0.0.0.0 0.0.0.0 0 0
outbound 11 permit 10.1.1.11 255.255.255.255 0 0
outbound 11 permit 10.1.1.12 255.255.255.255 0 0
outbound 11 permit 0.0.0.0 0.0.0.0 21 tcp
outbound 11 permit 10.3.3.3 255.255.255.255 143 tcp
```

This last **outbound** group lets hosts on the perimeter only access TCP ports 389 and 30303 and UDP port 53 (DNS).

Note The PIX Firewall drops DNS packets sent to UDP port 53 that have a packet size larger than 512 bytes.

Finally, the **apply** command statements set the **outbound** groups so that the permit and deny rules affect access to all external addresses.

```
outbound 13 deny 0.0.0.0 0.0.0.0 0 0
outbound 13 permit 0.0.0.0 0.0.0.0 389 tcp
outbound 13 permit 0.0.0.0 0.0.0.0 30303 tcp
outbound 13 permit 0.0.0.0 0.0.0.0 53 udp

apply (inside) 9 outgoing_src
apply (inside) 11 outgoing_src
apply (perim) 13 outgoing_src
```

Controlling Outbound Connections

The following example prevents all inside hosts from starting outbound connections:

```
outbound 1 deny 0 0 0
apply (inside) 1 outgoing_src
```

The **0 0 0** at the end of the command means all IP addresses (**0** is the same as **0.0.0.0**), with a 0.0.0.0 subnet mask and for all services (port value is zero).

Conversely, the following example permits all inside hosts to start connections to the outside (this is the default if an access list is not created):

```
outbound 1 permit 0 0 0
apply (inside) 1 outgoing_src
```

Controlling Inside Hosts' Access to Outbound Services

The following example prevents inside host 192.168.1.49 from accessing the World Wide Web (port 80):

```
outbound 11 deny 192.168.1.49 255.255.255.255 80 tcp
apply (inside) 11 outgoing_src
```

Controlling Inside Hosts' Access to Outside Servers

If your employees are spending too much time examining GIF images on a particular website with two web servers, you can use the following example to restrict this access:

```
outbound 12 deny 192.168.146.201 255.255.255.255 80 tcp
outbound 12 deny 192.168.146.202 255.255.255.255 80 tcp
apply (inside) 12 outgoing_dest
```

Using except Command Statements

An **except** command statement only provides exception to items with the same *list_ID*, as shown in the following example:

```
outbound 9 deny 0.0.0.0 0.0.0.0 0 0
outbound 9 except 10.100.0.0 255.255.0.0 23 tcp
outbound 9 except 0.0.0.0 0.0.0.0 53 udp
outbound 11 deny 0.0.0.0 0.0.0.0 0 0
outbound 11 permit 10.1.1.11 255.255.255.255 0 0
outbound 11 permit 10.1.1.12 255.255.255.255 0 0
outbound 11 permit 0.0.0.0 0.0.0.0 21 tcp
outbound 11 permit 10.3.3.3 255.255.255.255 143 tcp
outbound 13 deny 0.0.0.0 0.0.0.0 0 0
outbound 13 permit 0.0.0.0 0.0.0.0 389 tcp
outbound 13 permit 0.0.0.0 0.0.0.0 30303 tcp
outbound 13 permit 0.0.0.0 0.0.0.0 53 udp
```

In the preceding examples, the following two command statements work against other command statements in list 9 but not in lists 11 and 13:

```
outbound 9 except 10.100.0.0 255.255.0.0 23 tcp
outbound 9 except 0.0.0.0 0.0.0.0 53 udp
```

In the following example, the set of **deny**, **permit**, and **except** option command statements denies everybody from connecting to external hosts except for DNS queries and Telnet connections to hosts on 10.100.0.0. The host with IP address 10.1.1.11 is permitted outbound access, and has access to everywhere *except* to 10.100.0.0 via Telnet and anywhere to use DNS.

```
outbound   1 deny 0.0.0.0 0.0.0.0 0 tcp
outbound   1 permit 10.1.1.11 255.255.255.255 0 tcp
outbound   1 except 10.100.0.0 255.255.0.0 23 tcp
outbound   1 except 0.0.0.0 0.0.0.0 53 udp
apply (inside) outgoing_src
```

pager

Enable or disable screen paging.

[**no**] **pager** [**lines** *number*]

clear pager

show pager

Syntax Description

number	The number of lines before the "---more---" prompt appears. The minimum is **1**. Use **0** to disable paging.

Command Modes

Privileged mode.

Usage Guidelines

The **pager lines** command let you specify the number of lines in a page before the "---more---" prompt appears. The **pager** command enables display paging, and the **no pager** command disables paging and lets output display completely without interruption. If you set the **pager lines** command to some value and want to revert back to the default, enter the **pager** command without options. The **clear pager** command resets the number of lines in a page to 24.

When paging is enabled, the following prompt appears:

```
<--- more --->
```

The "---more---" prompt uses syntax similar to the UNIX **more** command:

- To view another screenful, press the Space bar.
- To view the next line, press the **Enter** key.
- To return to the command line, press the **q** key.

Use the **pager 0** command to disable paging.

Examples

The following example shows use of the **pager** command:

```
pixfirewall# pager lines 2
pixfirewall# ping inside 10.0.0.42
        10.0.0.42 NO response received -- 1010ms
        10.0.0.42 NO response received -- 1000ms
<--- more --->
```

password

Set password for Telnet access to the PIX Firewall console.

{**password** | **passwd**} *password* [**encrypted**]

clear {**password** | **passwd**}

show {**password** | **passwd**}

Syntax Description

encrypted	Specifies that the password you entered is already encrypted. The *password* you specify with the **encrypted** option must be 16 characters in length.
password	A case-sensitive password of up to 16 alphanumeric and special characters. Any character can be used in the password except a question mark and a space.

Command Modes

Privileged and configuration modes.

Usage Guidelines

The **password** command sets a password for Telnet access to the PIX Firewall console. The keyword **passwd** is also accepted as a shortened form of **password**. Additionally, the firewall configuration displays the password using the short form, **passwd.**

An empty password is changed into an encrypted string. However, any use of a **write** command displays or writes the passwords in encrypted form. Once passwords are encrypted, they are not reversible back to plain text. The **clear password** command resets the password to "cisco."

Note Store your new password in a manner consistent with your site's security policy. Once you change this password, you cannot view it again.

The **show password** command displays the Telnet password.

Examples

The following example shows use of the **password** command:

```
pixfirwall(config)# password watag00s1am
pixfirwall(config)# show passwd
passwd jMorNbK0514fadBh encrypted
```

Related Commands

enable	Configures enable passwords.
telnet	Adds Telnet access to the firewall console and sets the idle timeout.

pdm

These commands support communication between the PIX Firewall and a browser running the Cisco PIX Device Manager (PDM).

show pdm sessions

pdm disconnect *session_id*

pdm history enable

pdm history [view {all | 12h | 5d | 60m | 10m}] [snapshot] [feature {all | blocks | cpu | failover | ids | interface *if_name* **| memory | perfmon | xlates}] [pdmclient]**

pdm group *real_group_name associated_intf_name*

pdm group *ref_group_name ref_intf_name* **reference** *real_group_name*

pdm location ***ip_address netmask if_name***

pdm logging [*level* **[***messages***]]**

show pdm history

show pdm logging

show pdm sessions

clear pdm

Syntax Description

12h | 5d | 60m | 10m | all	Specifies the PDM history view to display: 12 hours (**12h**), 5 days (**5d**), 60 minutes (**60m**),10 minutes (**10m**), or **all** history contents in the PDM history buffer.
associated_intf_name	The name of the interface to which the specified object group is associated. This name must have been defined by the **nameif** command.
blocks	History for system buffers. Similar to output from the **show blocks** command.
clear pdm	Removes all locations, disables logging, and clears the PDM buffer. Internal PDM command.
cpu	History for CPU usage. Similar to output from the **show cpu usage** command.
failover	History for failover. Similar to output from the **show failover** command.
feature	This specifies to display history for a single feature (selected with one of the following). Otherwise, all of them are displayed.

group	Do not manually configure this command. PDM adds **pdm group** commands to the running configuration and uses them for internal purposes. This command is included in the documentation for informational purposes only.
history enable	Internal PDM command. Take a data sample and store the sample data to the PDM history buffer. The **no** version of this command disables PDM data sampling.
ids	History for IDS (Intrusion Detection System).
if_name	Specifies the interface name on which PDM resides.
ip_address	Specifies the host or network on which PDM resides.
level	Specifies the priority level of syslog messages displayed in the PDM **syslog** option.
location	Assists PDM with network topology discovery by associating an external network object with an interface. Note: The **pdm location** command does not control which host can launch PDM. See [**no**] **http** *ip_address* [*netmask*] [*if_name*] for this function. Do not manually configure this command. PDM adds **pdm group** commands to the running configuration and uses them for internal purposes. This command is included in the documentation for informational purposes only.
logging	Internal PDM command. Specifies the type and number of syslog messages displayed through the PDM **syslog** option.
memory	History for memory. Similar to output from the **show memory** command.
messages	Specifies the number of messages stored in the PDM buffer. Once the buffer is full, old messages will be discarded.
netmask	Specifies the network mask for the **pdm location** *ip_address*.
pdm	Specifies the Cisco PIX Device Manager.
pdm disconnect	Disconnects the specified PDM session from the PIX Firewall.
pdmclient	Displays the PDM history in PDM-display format.
perfmon	History for performance. Similar to output from the **show perfmon** command.
session_id	PDM session ID number available from the **show pdm sessions** command.
snapshot	Displays only the last PDM history data point.
real_group_name	The name of a PDM object group that contains real IP addresses.
ref_group_name	The name of an object group which contains network address translated (NATed) IP addresses of the object group specified by *real_group_name*.
ref_intf_name	The name of the interface from which the destination IP address of inbound traffic is network address translated (NATed). This name must have been defined by the **nameif** command.
xlates	History for translation slot information. Similar to output from the **show xlate** command.

Defaults

Default PDM syslog *level* is **0**. Default logging *messages* is **100** and the maximum is **512**.

Command Modes

Configuration mode.

Usage Guidelines

The **pdm disconnect** command and the **show pdm sessions** commands are accessible through the PIX Firewall command-line interface (CLI). The **show pdm sessions** command lists all the active PDM sessions connected to the PIX Firewall by a unique se*ssion_id*, beginning with session number **0**. The **pdm disconnect** command lets you disconnect a specific PDM session using its *session_id*.

The **show pdm history** command displays the contents of the PDM history buffer.

The **show pdm logging** command displays the contents of the PDM logging buffer (located within PDM). PDM syslog messages are stored separately from the PIX Firewall syslog messages. The **clear pdm logging** command clears the PDM log without disabling PDM logging.

The **clear pdm, pdm group, pdm history**, **pdm location, and pdm logging** commands may appear in your configuration, but they are designed to work as internal PDM-to-PIX Firewall commands accessible only to PDM.

The **pdm location** command associates an interface to an *ip_address /netmask* pair. Specifying a new pair replaces the old definition. The **clear pdm location** command removes all of the PDM locations.

Note Note: The **pdm location** command does not control which host can launch PDM. See [**no**] **http** *ip_address* [*netmask*] [*if_name*] for this function.

PDM location is not actually a PIX command, but rather a PDM bookkeeping command. When PDM opens it discovers the network topology surrounding the PIX from which it was launched. PDM then stores its discovered topology database in the PIX config file using **pdm location** commands to record ip address to interface associations. For example:

```
pdm location 10.1.1.1 255.255.255.255 inside
pdm location 10.1.1.2 255.255.255.255 inside
pdm location 10.1.3.0 255.255.255.0 inside
pdm location 10.1.2.0 255.255.255.0 outside
pdm location InsideRouter 255.255.255.255 inside
```

PDM rules are built on top of the network topology it can discover or has explicitly defined. Ideally, the topology is clearly defined first via the Host/Network and Network Object functions before policy Rules are applied.

You may use the CLI command **clear pdm location** to remove **pdm location** commands from your configuration, and it will not affect the operation of the PIX. However the next time PDM is run, it will again have to rediscover the network topology and update the configuration file with **pdm location** commands.

If you have an existing configuration before migrating to PDM, or use both the CLI and PDM to configure your PIX Firewall, PDM will derive much of the topology information from the current config file. For example:

```
static (inside,outside) 2.2.2.2 1.1.1.2 netmask 255.255.255.255 0 0
```

This command implies that **host 1.1.1.2** resides on the **inside** network.

Why is **pdm location** needed if PDM can derive or discover the topology information at runtime?

- The **static** command can be removed. If the location of **1.1.1.2** is not defined elsewhere in the config, the interface association will not be available to PDM. This can happen if you implicitly changed topology while editing an Access Rule or Translation Rule.

- PDM may not be able to resolve all the IP addresses shown in a configuration. For example, a PIX with three interfaces uses the CLI command **acl permit ip any 1.1.1.1** applied to **inside** interface. Where is **1.1.1.1, dmz** or **outside**? If you manually resolve **1.1.1.1** to the **outside** interface, for example, PDM will need to "remember" the interface to IP address association to allow Rules to be accurately displayed and edited.

The following example shows how to report the last data point in PDM-display format:

```
pix(config)# pdm history enable
pix(config)# show pdm history view 10m snapshot pdmclient
INTERFACE|outside|up|IBC|0|OBC|1088|IPC|0|OPC|0|IBR|17|OBR|
0|IPR|0|OPR|0|IERR|1|NB|0|RB|0|RNT|0|GNT|0|CRC|0|FRM|0|OR|
0|UR|0|OERR|0|COLL|0|LCOLL|0|RST|0|DEF|0|LCR|
0:PIXoutsideINTERFACE:METRIC_HISTORY|SNAP|IBR|VIEW|10|1952|
METRIC_HISTORY|SNAP|OBR|VIEW|10|64|METRIC_HISTORY|SNAP|IPR|
VIEW|10|17|METRIC_HISTORY|SNAP|OPR|VIEW|10|1|METRIC_HISTORY|
SNAP|IERR|VIEW|10|0|METRIC_HISTORY|SNAP|OERR|VIEW|10|0|
:PIXinsideINTERFACE:METRIC_HISTORY|SNAP|IBR|VIEW|10|0|
METRIC_HISTORY|SNAP|OBR|VIEW|10|64|METRIC_HISTORY|SNAP|IPR|
VIEW|10|0|METRIC_HISTORY|SNAP|OPR|VIEW|10|1|METRIC_HISTORY|
SNAP|IERR|VIEW|10|0|METRIC_HISTORY|SNAP|OERR|VIEW|10|0|
:PixSYS:METRIC_HISTORY|SNAP|MEM|VIEW|10|52662272|
METRIC_HISTORY|SNAP|BLK4|VIEW|10|1600|METRIC_HISTORY|SNAP|
BLK80|VIEW|10|400|METRIC_HISTORY|SNAP|BLK256|VIEW|10|998|
METRIC_HISTORY|SNAP|BLK1550|VIEW|10|676|METRIC_HISTORY|SNAP|
XLATES|VIEW|10|0|METRIC_HISTORY|SNAP|CONNS|VIEW|10|0|
METRIC_HISTORY|SNAP|TCPCONNS|VIEW|10|0|METRIC_HISTORY|SNAP|
UDPCONNS|VIEW|10|0|METRIC_HISTORY|SNAP|URLS|VIEW|10|0|
METRIC_HISTORY|SNAP|WEBSNS|VIEW|10|0|METRIC_HISTORY|SNAP|
TCPFIXUPS|VIEW|10|0|METRIC_HISTORY|SNAP|TCPINTERCEPTS|VIEW|
10|0|METRIC_HISTORY|SNAP|HTTPFIXUPS|VIEW|10|0|METRIC_HISTORY|
SNAP|FTPFIXUPS|VIEW|10|0|METRIC_HISTORY|SNAP|AAAAUTHENUPS|VIEW|
10|0|METRIC_HISTORY|SNAP|AAAAUTHORUPS|VIEW|10|0|METRIC_HISTORY|
SNAP|AAAACCOUNTS|VIEW|10|0|
```

The following example shows how to report the data, formatted for the PIX Firewall CLI:

```
pix(config)# pdm history enable
pix(config)# show pdm history view 10m snapshot
Available 4 byte Blocks:  [  10s] : 1600
Used 4 byte Blocks:  [  10s] : 0
Available 80 byte Blocks:  [  10s] : 400
Used 80 byte Blocks:  [  10s] : 0
Available 256 byte Blocks:  [  10s] : 500
Used 256 byte Blocks:  [  10s] : 0
Available 1550 byte Blocks:  [  10s] : 931
Used 1550 byte Blocks:  [  10s] : 385
Available 1552 byte Blocks:  [  10s] : 0
Used 1552 byte Blocks:  [  10s] : 0
Available 2560 byte Blocks:  [  10s] : 0
Used 2560 byte Blocks:  [  10s] : 0
Available 4096 byte Blocks:  [  10s] : 0
Used 4096 byte Blocks:  [  10s] : 0
Available 8192 byte Blocks:  [  10s] : 0
Used 8192 byte Blocks:  [  10s] : 0
Available 16384 byte Blocks:  [  10s] : 0
Used 16384 byte Blocks:  [  10s] : 0
Available 65536 byte Blocks:  [  10s] : 0
Used 65536 byte Blocks:  [  10s] : 0
CPU Utilization:  [  10s] : 0
IP Options Bad:  [  10s] : 0
Record Packet Route:  [  10s] : 0
IP Options Timestamp:  [  10s] : 0
Provide s,c,h,tcc:  [  10s] : 0
Loose Source Route:  [  10s] : 0
SATNET ID:  [  10s] : 0
Strict Source Route:  [  10s] : 0
IP Fragment Attack:  [  10s] : 0
Impossible IP Attack:  [  10s] : 0
IP Teardrop:  [  10s] : 0
```

```
ICMP Echo Reply:  [  10s] : 0
ICMP Unreachable:  [  10s] : 0
ICMP Source Quench:  [  10s] : 0
ICMP Redirect:  [  10s] : 0
ICMP Echo Request:  [  10s] : 0
ICMP Time Exceeded:  [  10s] : 0
ICMP Parameter Problem:  [  10s] : 0
ICMP Time Request:  [  10s] : 0
ICMP Time Reply:  [  10s] : 0
ICMP Info Request:  [  10s] : 0
ICMP Info Reply:  [  10s] : 0
ICMP Mask Request:  [  10s] : 0
ICMP Mask Reply:  [  10s] : 0
Fragmented ICMP:  [  10s] : 0
Large ICMP:  [  10s] : 0
Ping of Death:  [  10s] : 0
No Flags:  [  10s] : 0
SYN & FIN Only:  [  10s] : 0
FIN Only:  [  10s] : 0
FTP Improper Address:  [  10s] : 0
FTP Improper Port:  [  10s] : 0
Bomb:  [  10s] : 0
Snork:  [  10s] : 0
Chargen:  [  10s] : 0
DNS Host Info:  [  10s] : 0
DNS Zone Transfer:  [  10s] : 0
DNS Zone Transfer High Port:  [  10s] : 0
DNS All Records:  [  10s] : 0
Port Registration:  [  10s] : 0
Port Unregistration:  [  10s] : 0
RPC Dump:  [  10s] : 0
Proxied RPC:  [  10s] : 0
ypserv Portmap Request:  [  10s] : 0
ypbind Portmap Request:  [  10s] : 0
yppasswd Portmap Request:  [  10s] : 0
ypupdated Portmap Request:  [  10s] : 0
ypxfrd Portmap Request:  [  10s] : 0
mountd Portmap Request:  [  10s] : 0
rexd Portmap Request:  [  10s] : 0
rexd Attempt:  [  10s] : 0
statd Buffer Overflow:  [  10s] : 0
Input KByte Count:  [  10s] : 41804
Output KByte Count:  [  10s] : 526456
Input KPacket Count:  [  10s] : 364
Output KPacket Count:  [  10s] : 450
Input Bit Rate:  [  10s] : 0
Output Bit Rate:  [  10s] : 0
Input Packet Rate:  [  10s] : 0
Output Packet Rate:  [  10s] : 0
Input Error Packet Count:  [  10s] : 0
No Buffer:  [  10s] : 0
Received Broadcasts:  [  10s] : 90076
Runts:  [  10s] : 0
Giants:  [  10s] : 0
CRC:  [  10s] : 0
Frames:  [  10s] : 0
Overruns:  [  10s] : 0
Underruns:  [  10s] : 0
Output Error Packet Count:  [  10s] : 0
Collisions:  [  10s] : 8895
LCOLL:  [  10s] : 0
Reset:  [  10s] : 0
Deferred:  [  10s] : 3138
Lost Carrier:  [  10s] : 0
```

```
Hardware Input Queue:  [  10s] : 128
Software Input Queue:  [  10s] : 0
Hardware Output Queue:  [  10s] : 0
Software Output Queue:  [  10s] : 0
Input KByte Count:  [  10s] : 61835
Output KByte Count:  [  10s] : 26722
Input KPacket Count:  [  10s] : 442
Output KPacket Count:  [  10s] : 418
Input Bit Rate:  [  10s] : 0
Output Bit Rate:  [  10s] : 0
Input Packet Rate:  [  10s] : 0
Output Packet Rate:  [  10s] : 0
Input Error Packet Count:  [  10s] : 0
No Buffer:  [  10s] : 0
Received Broadcasts:  [  10s] : 308607
Runts:  [  10s] : 0
Giants:  [  10s] : 0
CRC:  [  10s] : 0
Frames:  [  10s] : 0
Overruns:  [  10s] : 0
Underruns:  [  10s] : 0
Output Error Packet Count:  [  10s] : 0
Collisions:  [  10s] : 0
LCOLL:  [  10s] : 0
Reset:  [  10s] : 0
Deferred:  [  10s] : 2
Lost Carrier:  [  10s] : 707
Hardware Input Queue:  [  10s] : 128
Software Input Queue:  [  10s] : 0
Hardware Output Queue:  [  10s] : 0
Software Output Queue:  [  10s] : 0
Available Memory:  [  10s] : 45293568
Used Memory:  [  10s] : 21815296
Xlate Count:  [  10s] : 0
Connection Count:  [  10s] : 0
TCP Connection Count:  [  10s] : 0
UDP Connection Count:  [  10s] : 0
URL Filtering Count:  [  10s] : 0
URL Server Filtering Count:  [  10s] : 0
TCP Fixup Count:  [  10s] : 0
TCP Intercept Count:  [  10s] : 0
HTTP Fixup Count:  [  10s] : 0
FTP Fixup Count:  [  10s] : 0
AAA Authentication Count:  [  10s] : 0
AAA Authorzation Count:  [  10s] : 0
AAA Accounting Count:  [  10s] : 0
Current Xlates:  [  10s] : 0
Max Xlates:  [  10s] : 0
ISAKMP SAs:  [  10s] : 0
IPSec SAs:  [  10s] : 0
L2TP Sessions:  [  10s] : 0
L2TP Tunnels:  [  10s] : 0
PPTP Sessions:  [  10s] : 0
PPTP Tunnels:  [  10s] : 0
```

Related Commands

setup	Preconfigures the firewall through interactive prompts.

perfmon

View performance information.

perfmon verbose

perfmon interval seconds

perfmon quiet

perfmon settings

show perfmon

Syntax Description

interval *seconds*	Specify the number of seconds the performance display is refreshed on the console. The default is 120 seconds.
quiet	Disable performance monitor displays.
settings	Displays the interval and whether it is quiet or verbose.
verbose	Enable displaying performance monitor information at the PIX Firewall console.

Command Modes

Privileged mode.

Usage Guidelines

The **perfmon** command lets you monitor the PIX Firewall unit's performance. Use the **show perfmon** command to view the information immediately. Use the **perfmon verbose** command to display the information every two minutes continuously. Use the **perfmon interval** *seconds* command with the **perfmon verbose** command to display the information continuously every number of seconds you specify.

Use the **perfmon quiet** command to disable the display.

The **show perfmon** command displays PIX Firewall performance information. (However, this command output does not display in a Telnet console session.)

An example of the performance information follows:

PERFMON STATS:	Current	Average
Xlates	33/s	20/s
Connections	110/s	10/s
TCP Conns	50/s	42/s
WebSns Req	4/s	2/s
TCP Fixup	20/s	15/s
HTTP Fixup	5/s	5/s
FTP Fixup	7/s	4/s
AAA Authen	10/s	5/s
AAA Author	9/s	5/s
AAA Account	3/s	3/s

This information lists the number of translations, connections, Websense requests, address translations (called "fixups"), and AAA transactions that occur each second.

Examples

The following commands display the performance monitor statistics every 30 seconds on the PIX Firewall console:

```
perfmon interval 30
perfmon verbose
```

ping

Determine if other IP addresses are visible from the PIX Firewall.

ping [*if_name*] *ip_address*

Syntax Description

if_name	The internal or external network interface name. The address of the specified interface is used as the source address of the ping.
ip_address	The IP address of a host on the inside or outside networks.

Command Modes

Privileged mode.

Usage Guidelines

The **ping** command determines if the PIX Firewall has connectivity or if a host is available on the network. The command output shows if the response was received; that is, that a host is participating on the network. If a host is not responding, **ping** displays "NO response received." Use the **show interface** command to ensure that the PIX Firewall is connected to the network and is passing traffic.

If you want internal hosts to be able to ping external hosts, you must create an ICMP **access-list** command statement for echo reply; for example, to give ping access to all hosts, use the **access-list** *acl_grp* **permit icmp any any** command and bind the **access-list** command statement to the interface you want to test using an **access-group** command statement.

If you are pinging through PIX Firewall between hosts or routers, but the pings are not successful, use the **debug icmp trace** command to monitor the success of the ping. If pings are both inbound and outbound, they are successful.

The PIX Firewall **ping** command no longer requires an interface name. If an interface name is not specified, PIX Firewall checks the routing table to find the address you specify. You can specify an interface name to indicate through which interface the ICMP echo requests are sent.

An example of the usage follows:

```
ping 10.0.0.1
         10.0.0.1 response received -- 10ms
         10.0.0.1 response received -- 10ms
         10.0.0.1 response received -- 0ms
```

Or you can still enter the command specifying the interface:

```
ping outside 10.0.0.1
         10.0.0.1 response received -- 10ms
         10.0.0.1 response received -- 10ms
```

10.0.0.1 response received -- 0ms

Examples In the following example, the **ping** command makes three attempts to reach an IP address:

```
ping 192.168.42.54
        192.168.42.54 response received -- 0Ms
        192.168.42.54 response received -- 0Ms
        192.168.42.54 response received -- 0Ms
```

prefix-list

Configures a prefix list for Area Border Router (ABR) type 3 link-state advertisement (LSA) filtering (to be used in OSPF routing areas).

[**no**] **prefix-list** *list_name* [**seq** *seq_number*] {**permit | deny** *prefix* **/** *len*} [**ge** *min_value*] [**le** *max_value*]

[**no**] **prefix-list sequence-number**

prefix-list *list_name* **description** *text*

Syntax Description

/	A required separator between the *prefix* and *len* values.
deny	Denies access for a matching condition.
ge	Applies the *min_value* to the range specified.
le	Applies the *max_value* to the range specified.
len	The network length (in bits) of the network mask, from 0 to 32.
list_name	The name of the prefix list. The *list_name* and *seq_number* together must be less than 64 characters combined.
max_value	Specifies the greater value of a range (the “to” portion of the range description). Ranges values can be from 0 to 32.
min_value	Specifies the lesser value of a range (the “from” portion of the range description). Ranges values can be from 0 to 32.
permit	Permits access for a matching condition.
prefix	The network number.
seq *seq_number*	Specifies the sequence number for the prefix list entry, from 1 to 4294967295. However, the *list_name* and *seq_number* together must be less than 64 characters combined.
sequence-number	Enables the generation of sequence numbers for entries in an OSPF prefix list.
text	The text of the description, with a maximum of 80 characters.

Defaults None.

Command Modes Configuration mode.

Usage Guidelines

The **prefix-list** commands are Area Border Router (ABR) type 3 link-state advertisement (LSA) filtering commands. ABR type 3 LSA filtering extends the capability of an ABR that is running OSPF to filter type 3 LSAs between different OSPF areas. This filtering is based on a prefix list defined by you, using the **prefix-list** commands. Once configured, only the specified prefixes are sent from one area to another area, and all other prefixes are restricted to their OSPF area. This type of area filtering can be applied to traffic going into or coming out of an OSPF area, or to both the incoming and outgoing traffic for that area.

To create an entry in a prefix list, use the **prefix-list** *list_name* command. To delete the entry, use the **no prefix-list** *list_name* command.

Use the **prefix-list** *list_name* **description** *text* command to add a text description to the prefix list name. To remove the text description, use the **no prefix-list** *list_name* **description** *text* command.

The **prefix-list** *list_name* **seq** *seq_number* command designates sequence numbers for entries in a prefix list.

Use the **prefix-list sequence-number** command to enable the generation of sequence numbers for entries in a OSPF prefix list.

Examples

The following example shows how to configure a prefix list:

```
pixfirewall(config)# prefix-list t-prelist permit 5/0001
pixfirewall(config)# show prefix-list
prefix-list t-prelist seq 5 permit 0.0.0.0/1
```

Related Commands

area filter-list	A subcommand to the **router ospf** command that uses the prefix list that you configure with the **prefix-list** command.
route-map	Creates a route map for redistributing routes from one routing protocol to another. Used in configuring OSPF routing on the firewall.
router ospf	Configures global parameters for the OSPF routing processes on the firewall, and enables or disables OSPF routing through the firewall.
routing interface	Configures interface-specific OSPF routing parameters.

privilege

Configures or displays command privilege levels.

[**no**] **privilege** [**show** | **clear** | **configure**] **level** *level* [**mode** *enable* | *configure*] **command** *command*

show curpriv

show privilege [**all** | **command** *command* | **level** *level*]

Syntax Description

clear	Sets the privilege level for the **clear** command corresponding to the command specified.
command	The command to allow. (Use the **no** command form to disallow.)
command	The command on which to set the privilege level.

configure	Sets the privilege level for the **configure** command corresponding to the command specified.
configure	For commands with both enable and configure modes, this indicates that the level is for the configure mode of the command.
curpriv	Displays the current privilege level.
detail	Displays privilege debugging information.
enable	For commands with both enable and configure modes, this indicates that the level is for the enable mode of the command.
level	The privilege level, from 0 to 15. (Lower numbers are lower privilege levels.)
level	Specifies the privilege level.
show	Sets the privilege level for the **show** command corresponding to the command specified.

Command Modes

Configuration mode.

Usage Guidelines

The privilege command sets user-defined privilege levels for PIX Firewall commands. This is especially useful for setting different privilege levels for related configuration, show, and clear commands. However, be sure to verify privilege level changes in your commands with your security policies before implementing the new privilege levels.

When commands have privilege levels set, and users have privilege levels set, then the two are compared to determine if a given user can execute a given command. If the user's privilege level is lower than the privilege level of the command, the user is prevented from executing the command. This is modeled after Cisco IOS software.

To change between privilege levels, use the **login** command to access another privilege level and the appropriate **logout**, **exit**, or **quit** command to exit that level.

Note Your **aaa authentication** and **aaa authorization** commands need to include any new privilege levels you define before you can use them in your AAA server configuration.

The **show curpriv** command displays the current privileges for a user.

The **show privilege** [**all** | **command** *command* | **level** *level*] command displays the privileges for a command or set of commands.

Examples

You can set the privilege level “5” for an individual user as follows:

```
username intern1 password pass1 privilege 5
```

You can also define a set of **show** commands with the privilege level “5” as follows:

```
level:

privilege show level 5 command alias
privilege show level 5 command apply
privilege show level 5 command arp
privilege show level 5 command auth-prompt
privilege show level 5 command blocks
```

The following examples show output from the **show curpriv** command when a user named **enable_15** is at different privilege levels. **Username** indicates the name the user entered when he or she logged in, **P_PRIV** indicates that the user has entered the **enable** command, and **P_CONF** indicates the user has entered the **config terminal** command.

```
pixfirewall(config)# show curpriv
Username : enable_15
Current privilege level : 15
Current Mode/s : P_PRIV P_CONF
pixfirewall(config)# exit

pixfirewall# show curpriv
Username : enable_15
Current privilege level : 15
Current Mode/s : P_PRIV
pixfirewall# exit

pixfirewall> show curpriv
Username : enable_1
Current privilege level : 1
Current Mode/s : P_UNPR
pixfirewall>
```

The following is an example of applying a privilege level of 11 to a complete AAA authorization configuration:

```
privilege configure level 11 command aaa
privilege configure level 11 command aaa-server
privilege configure level 11 command access-group
privilege configure level 11 command access-list
privilege configure level 11 command activation-key
privilege configure level 11 command age
privilege configure level 11 command alias
privilege configure level 11 command apply
```

Related Commands

aaa authentication	Enable, disable, or view LOCAL, TACACS+, or RADIUS user authentication, on a server designated by the **aaa-server** command, or PDM user authentication
login	Logs into a new privilege level.
object-group	Create an object group for use in other commands, such as **access-list** statements.
username	Configures local user authentication database.

quit

Exit configuration or privileged mode.

quit

Syntax Description

quit	Exits the current privilege level or mode.

Command Modes All modes.

Usage Guidelines Use the **quit** command to exit configuration or privileged mode.

Examples The following example shows use of the **quit** command:

```
pixfirewall(config)# quit
pixfirewall# quit
pixfirewall>
```

reload

Reboot and reload the configuration.

reload [**noconfirm**]

Syntax Description

noconfirm	Permits the PIX Firewall to reload without user confirmation.
reload	Reboot and reload configuration.

Command Modes Privileged mode.

Usage Guidelines The **reload** command reboots the PIX Firewall and reloads the configuration from a bootable floppy disk or, if a diskette is not present, from Flash memory.

The PIX Firewall does not accept abbreviations to the keyword **noconfirm**.

You are prompted for confirmation before starting with "Proceed with reload?".
Any response other than **n** causes the reboot to occur.

Note Configuration changes not written to Flash memory are lost after reload. Before rebooting, store the current configuration in Flash memory with the **write memory** command.

Examples The following example shows use of the **reload** command:

```
reload
Proceed with reload?  [confirm] y

Rebooting...

PIX Bios V2.7
...
```

rip

Change Routing Information Protocol (RIP) settings.

[**no**] **rip** *if_name* **default | passive** [**version** [**1 | 2**]] [**authentication** [**text | md5** *key* (*key_id*)]]

debug rip [*if_name*]

clear rip

show rip [*if_name*]

Syntax Description

authentication	Enable RIP Version 2 authentication.
default	Broadcast a default route on the interface.
if_name	The internal or external network interface name.
key	Key to encrypt RIP updates. This value must be the same on the routers and any other device *that provides RIP Version 2 updates*. The *key* is a text string of up to 16 characters in length.
key_id	Key identification value. The *key_id* can be a number from 1 to 255. Use the same *key_id* that is in use on the routers and any other device that provides RIP Version 2 updates.
md5	Send RIP updates using MD5 encryption.
passive	Enable passive RIP on the interface. The PIX Firewall listens for RIP routing broadcasts and uses that information to populate its routing tables.
text	Send RIP updates as clear text (not recommended).
version	RIP version. Use **version 2** for RIP update encryption. Use **version 1** to provide backward compatibility with the older version.

Command Modes

Configuration mode.

Usage Guidelines

The **rip** command enables IP routing table updates from received Routing Information Protocol (RIP) broadcasts. Use the **no rip** command to disable the PIX Firewall IP routing table updates. The default is to enable IP routing table updates. If you specify RIP Version 2, you can encrypt RIP updates using MD5 encryption.

The **clear rip** command removes all the **rip** commands from the configuration.

Ensure that the key and key_id values are the same as in use on any other device in your network that makes RIP Version 2 updates.

The PIX Firewall cannot pass RIP updates between interfaces.

When RIP Version 2 is configured in passive mode with PIX Firewall software Version 5.3 and higher, the PIX Firewall accepts RIP Version 2 multicast updates with an IP destination of 224.0.0.9. For RIP Version 2 default mode, the PIX Firewall will transmit default route updates using an IP destination of 224.0.0.9. Configuring RIP Version 2 registers the multicast address 224.0.0.9 on the respective interface to be able to accept multicast RIP Version 2 updates.

Only Intel 10/100 and Gigabit interfaces support multicasting.

When the RIP Version 2 commands for an interface are removed, the multicast address is unregistered from the interface card.

Examples

The following is sample output from the Version 1 **show rip** and **rip inside default** commands:

```
show rip
rip outside passive
no rip outside default
rip inside passive
no rip inside default

rip inside default
show rip
rip outside passive
no rip outside default
rip inside passive
rip inside default
```

The next example combines Version 1 and Version 2 commands and shows listing the information with the **show rip** command after entering the RIP commands that do the following:

- Enable Version 2 passive RIP using MD5 authentication on the outside interface to encrypt the key used by the PIX Firewall and other RIP peers, such as routers.
- Enable Version 1 passive RIP listening on the inside interface of the PIX Firewall.
- Enable Version 2 passive RIP listening on the **dmz** interface of the PIX Firewall.

```
rip outside passive version 2 authentication md5 thisisakey 2
rip outside default version 2 authentication md5 thisisakey 2
rip inside passive
rip dmz passive version 2

show rip
rip outside passive version 2 authentication md5 thisisakey 2
rip outside default version 2 authentication md5 thisisakey 2
rip inside passive version 1
rip dmz passive version 2
```

The next example shows how use of the **clear rip** command clears all the previous **rip** commands from the current configuration:

```
clear rip
show rip
```

The following example shows use of the Version 2 feature that passes the encryption key in text form:

```
rip out default version 2 authentication text thisisakey 3
show rip
rip outside default version 2 authentication text thisisakey 3
```

route

Enter a static or default route for the specified interface.

[**no**] **route** *if_name ip_address netmask gateway_ip* [*metric*]

clear route [*if_name ip_address* [*netmask gateway_ip*]]

show route

Syntax Description

gateway_ip	Specify the IP address of the gateway router (the next hop address for this route).
if_name	The internal or external network interface name.
ip_address	The internal or external network IP address. Use **0.0.0.0** to specify a default route. The **0.0.0.0** IP address can be abbreviated as **0**.
metric	Specify the number of hops to *gateway_ip*. If you are not sure, enter **1**. Your network administrator can supply this information or you can use a **traceroute** command to obtain the number of hops. The default is **1** if a metric is not specified.
netmask	Specify a network mask to apply to *ip_address*. Use **0.0.0.0** to specify a default route. The **0.0.0.0** *netmask* can be abbreviated as **0**.

Command Modes

Configuration mode.

Usage Guidelines

Use the **route** command to enter a default or static route for an interface. To enter a default route, set *ip_address* and *netmask* to **0.0.0.0,** or the shortened form of **0**. All routes entered using the **route** command are stored in the configuration when it is saved. The **clear route** command removes **route** command statements from the configuration that do not contain the CONNECT keyword.

Create static routes to access networks connected outside a router on any interface. The effect of a static route is like stating "to send a packet to the specified network, give it to this router." For example, PIX Firewall sends all packets destined to the 192.168.42.0 network through the 192.168.1.5 router with this static **route** command statement.

```
route dmz 192.168.42.0 255.255.255.0 192.168.1.5 1
```

The routing table automatically specifies the IP address of a PIX Firewall interface in the **route** command. Once you enter the IP address for each interface, PIX Firewall creates a **route** statement entry that is not deleted when you use the **clear route** command.

Note As of PIX Firewall Version 6.3(2), the **show route** command displays the route information only for active routes. Routes that are configured on interfaces and administratively or physically shut down do not display with the **show route** command.

If the **route** command statement uses the IP address from one interface of the PIX Firewall unit as the gateway IP address, PIX Firewall will ARP for the destination IP address in the packet instead of ARPing for the gateway IP address.

The following steps show how PIX Firewall handles routing:

Step 1 PIX Firewall receives a packet from the inside interface destined to IP address X.

Step 2 Because a default route is set to itself, PIX Firewall sends out an ARP for address X.

Step 3 Any Cisco router on the outside interface LAN which has a route to address X (Cisco IOS software has proxy ARP enabled by default) replies back to the PIX Firewall with its own MAC address as the next hop.

Step 4 PIX Firewall sends the packet to router (just like a default gateway).

Step 5 PIX Firewall adds the entry to its ARP cache for IP address X with the MAC address being that of the router.

- The CONNECT route entry is supported. (This identifier appears when you use the **show route** command.) The CONNECT identifier is assigned to an interface's local network and the interface IP address, which is in the IP local subnet. PIX Firewall will ARP for the destination address. The CONNECT identifier cannot be removed, but changes when you change the IP address on the interface.
- If you enter duplicate routes with different metrics for the same gateway, PIX Firewall changes the metric for that route and updates the metric for the route.

  For example, if the following command statement is in the configuration:

  ```
  route inside 10.0.0.0 255.0.0.0 10.0.0.2 2 OTHER
  ```

  If you enter the following statement:

  ```
  route inside 10.0.0.0 255.0.0.0 10.0.0.2 3
  ```

  PIX Firewall converts the command statement to the following:

  ```
  route inside 10.0.0.0 255.0.0.0 10.0.0.2 3 OTHER
  ```

Examples

Specify one default **route** command statement for the outside interface, which in this example is for the router on the outside interface that has an IP address of 209.165.201.1:

```
route outside 0 0 209.165.201.1 1
```

For static routes, if two networks, 10.1.2.0 and 10.1.3.0 connect via a hub to the dmz1 interface router at 10.1.1.4, add these static **route** command statements to provide access to the networks:

```
route dmz1 10.1.2.0 255.0.0.0 10.1.1.4 1
route dmz1 10.1.3.0 255.0.0.0 10.1.1.4 1
```

route-map

Defines the conditions for redistributing routes from one routing protocol into another. (Used in configuring OSPF routing on the firewall.) OSPF routing is not supported on the PIX 501.

[**no**] **route-map** *map_tag* [**permit** | **deny**] [*seq_num*]

show route-map [*map_tag*]

Subcommands to the **route-map** command:

[**no**] **match** [**interface** *interface_name* | **metric** *metric_value* | **ip address** *acl_id* | **route-type** {**local** | **internal** | [**external** [**type-1** | **type-2**]]}| **nssa-external** [**type-1** | **type-2**] | **ip next-hop** *acl_id* | **ip route-source** *acl_id*]

[**no**] **set metric** *value*

[**no**] **set metric-type** {**type-1** | **type-2** | **internal** | **external**}

[**no**] **set ip next-hop** *ip-address* [*ip-address*]

Syntax Description

acl_id	The name of an ACL. The **match ip next-hop** and **match ip route-source** commands can accept more than one *acl_id*. That is, they accept *acl_id* [*...acl_id*].
deny	If the match criteria are met for the route map and the **deny** option is specified, the route is not redistributed.
external	The OSPF metric routes external to a specified autonomous system.
interface_name	The name of the interface.
internal	Routes that are internal to a specified autonomous system.
ip next-hop *ip-address* [*ip-address*]	Indicates where to output packets that pass a match clause of the route map.
ip route-source	Redistributes routes that have been advertised by routers and access servers at the address specified by the *acl_id*.
local	Specifies a preference value for the autonomous system path.
map_tag	The text for the route map tag, meant to define a meaningful name for the route map, up to 58 characters in length. Multiple route maps may share the same map tag name.
metric_value	A metric value, from 0 to 2147483647.
nssa-external [**type-1** | **type-2**]	The OSPF metric type for routes that are external to a not-so-stubby area (NSSA), either type 1 or 2. The default is type 2.
permit	If the match criteria are met for this route map, and the **permit** option is specified, the route is redistributed as controlled by the set actions. If the match criteria are not met, and the **permit** keyword is specified, the next route map with the same *map_tag* is tested. If a route passes none of the match criteria for the set of route maps sharing the same name, it is not redistributed by that set. The **permit** option is the default.
seq_num	If there are any route maps with the same *map_tag*, then you must also specify a *seq_num* for the route-maps to differentiate between them. The *seq_num* can be any number from 0 to 65535. Otherwise, no *seq_num* needs to be specified. A default value of 10 is assigned to the first route map if no *seq_num* is specified. If given in the **no route-map** *map_tag seq_num* command, *seq_num* is the route map to be deleted.
type-1 | type-2	The OSPF metric routes external to a specified autonomous system, either type 1 or 2. The default is type 2.

Defaults

The **permit** option is the default for the **route-map** command.

Command Modes

The **route-map** command is available in configuration mode.

The **show route-map** command is available in privileged mode.

Usage Guidelines

To define the conditions for redistributing routes from one routing protocol into another, use the **route-map** *map_tag* command and the **match** and **set route-map** configuration commands. To delete an entry, use the **no route-map** *map_tag* command.

set metric *value*

To set the metric value for a routing protocol, use the **set metric** *value* subcommand. To return to the default metric value, use the **no set metric** *value* subcommand. In this context, the *value* is an integer from -2147483647 to 2147483647.

set metric-type {**type-1 | type-2**}

To set the metric type for the destination routing protocol, use the **set metric-type** {**type-1 | type-2**} subcommand. To return to the default, use the **no set metric-type** {**type-1 | type-2**} subcommand.

set ip next-hop *ip-address*

To indicate where to output packets that pass a match clause of a route map, use the **set ip next-hop** *ip_address* subcommand. To delete an entry, use the **no set ip next-hop** *ip-address* subcommand. In this context, *ip_address* is the IP address of the next hop to which to output packets. It must be the address of an adjacent router.

Examples

The following example show how to configure a route map for use in OSPF routing:

```
pixfirewall(config)# route-map maptag1 permit 8
pixfirewall(config-route-map)# set metric 5
pixfirewall(config-route-map)# match metric 5
pixfirewall(config-route-map)# set metric-type type-2
pixfirewall(config-route-map)# show route-map
route-map maptag1 permit 8
  set metric 5
  set metric-type type-2
  match metric 5
pixfirewall(config-route-map)# exit
pixfirewall(config)#
```

Related Commands

prefix-list	Configures a prefix list to be used for OSPF routing.
router ospf	Configures global parameters for the OSPF routing processes on the firewall, and enables or disables OSPF routing through the firewall.
routing interface	Configures interface-specific OSPF routing parameters.

router ospf

Configures global parameters for the OSPF routing processes on the firewall, and enables or disables OSPF routing through the firewall. (Use the **routing interface** command for interface-specific OSPF configration.) OSPF routing is not supported on the PIX 501.

[**no**] **router ospf** *pid*

show router ospf *pid*

Subcommands to the **router ospf** command:

[**no**] **area** *area_id*

[**no**] **area** *area_id* **authentication** [**message-digest**]

[**no**] **area** *area_id* **default-cost** *cost*

[**no**] **area** *area_id* **filter-list prefix** {*prefix_list_name* **in | out**}

[**no**] **area** *area_id* **nssa** [**no-redistribution**] [**default-information-originate** [**metric-type 1 | 2**] [**metric** *metric_value*]]

[**no**] **area** *area_id* **range** *ip_address netmask* [**advertise | not-advertise**]

area *area_id* **stub** [**no-summary**]

[**no**] **area** *area_id* **virtual-link** *router_id* [**authentication** [**message-digest | null**]] [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [**transmit-delay** *seconds*] [**dead-interval** *seconds*] [**authentication-key** *password*] [**message-digest-key** *id* **md5** *password*]

[**no**] **compatible rfc1583**

default-information originate [**always**] [**metric** *metric_value*] [**metric-type** {**1 | 2**}] [**route-map** *map_name*]

[**no**] **distance ospf** [**intra-area** *d1*][**inter-area** *d2*][**external** *d3*]

[**no**] **ignore lsa mospf**

[**no**] **log-adj-changes** [**detail**]

[**no**] **network** *prefix ip_address netmask* **area** *area_id*

[**no**] **redistribute** {**static | connected**} [**metric** *metric_value*] [**metric-type** *metric_type*] [**route-map** *map_name*] [**tag** *tag_value*] [**subnets**]

[**no**] **redistribute ospf** *pid* [**match** {**internal | external** [**1|2**] **| nssa-external** [**1|2**]}] [**metric** *metric_value* **] [metric-type** *metric_type***] [route-map** *map_name***] [tag** *tag_value***] [subnets**]

[**no**] **router-id** *ip_address*

[**no**] **summary-address** *addr netmask* [**not-advertise**] [**tag** *tag_value*]

[**no**] **timers** {**spf** *spf_delay spf_holdtime* **| lsa-group-pacing** *seconds*}

Syntax Description

addr	The value of the summary address designated for a range of addresses.
advertise	Sets the address range status to advertise and generates a Type 3 summary link-state advertisements (LSA).
area *area_id*	Configures a regular OSPF area.
area_id	For all contexts, *area_id* can be specified as either a decimal value or as an IP address. The ID of the area that is to be associated with the OSPF address range. If you intend to associate areas with IP subnets, you can specify a subnet address as the *area_id*. When used in the context of authentication, *area_id* is the identifier of the area on which authentication is to be enabled. When using a cost context, *area_id* is the identifier for the stub or NSSA. When used in the context of a prefix list, *area_id* is the identifier of the area on which filtering is configured. When used in a stub area or not-so-stubby area (NSSA) context, *area_id* is the identifier for the stub or NSSA area. When used in the context of an area range, *area_id* is the identifier of the area at whose boundary to summarize routes.
authentication	(Optional) Specifies the authentication type.
compatible	Runs OSPF in RFC 1583 compatible mode.
cost	The cost for the default summary route used for a stub or NSSA, from 0 to 65535. The default value for *cost* is 1.
d1, *d2*, and *d3*	The distance for different area route types. The default for *d1*, *d2*, and *d3* is 110.
default-information	Distributes a default route according to the parameters specified.
default-information -originate	Used to generate a Type 7 default in the NSSA area. This keyword only takes effect on an NSSA ABR or an NSSA Autonomous System Boundary Router (ASBR).
distance	Configures administrative distances for the OSPF process.
external	Sets the distance for routes from other routing domains, learned by redistribution.
external 1 | 2	The OSPF metric routes external to a specified autonomous system, either type 1 or 2. The default is type 2.
ignore	Supresses syslog for receipt of type 6 Multicast OSPF LSAs.
in	Applies the configured prefix list to prefixes advertised inbound to the specified area.
inter-area	Sets the distance for all routes from one area to another area.
internal	Routes that are internal to a specified autonomous system.
ip_address	The router ID in IP address format.
log-adj-changes	Logs OSPF adjacency changes.
lsa-group-pacing *seconds*	The interval at which OSPF link-state advertisements (LSAs) are collected into a group and refreshed, checksummed, or aged, from 10 to 1800 seconds. The default value is 240 seconds.
map_name	The name of the route map to apply.

message-digest	(Optional) Enables Message Digest 5 (MD5) authentication on the area specified by the *area_id*.
metric *metric_value*	Specifies the OSPF default metric value, from 0 to 16777214.
netmask	An IP address mask, or IP subnet mask used for a summary route.
network	Adds/removes interfaces to/from the OSPF routing process.
no-redistribution	When the OSPF router is an NSSA Area Border Router (ABR) and you want the **redistribute** command to import routes only into the normal areas, and not into the NSSA area, use this option.
no-summary	Prevents an Area Border Router (ABR) from sending summary link-state advertisements into the stub area.
not-advertise	Sets the address range status to DoNotAdvertise. The Type 3 summary LSA is suppressed, and the component networks remain hidden from other networks. In the **summary-address** command, **not-advertise** suppresses routes that match the specified prefix/mask pair.
nssa-external 1 | 2	The OSPF metric type for routes that are external to a not-so-stubby area (NSSA), either type 1 or 2. The default is type 2.
null	(Optional) Specifies that no authentication is used. Overrides password or message digest authentication if configured for the OSPF area.
out	Applies the configured prefix list to prefixes advertised outbound from the specified area.
pid	Internally used identification parameter for an OSPF routing process. You assign it locally on the firewall, and it can be from 1 to 65535. A unique value must be assigned for each OSPF routing process. PIX Firewall software Version 6.3 supports a maximum of two (2) OSPF processes.
prefix	Indicates that a prefix list is used. (Prefix lists are configured with the **prefix-list** command.)
prefix	An IP address.
prefix_list_name	Name of a prefix list.
redistribute	Configures redistribution between OSPF processes according to the parameters specified.
router-id	Configures the router ID for an OSPF process.
seconds	The interval at which OSPF link-state advertisements (LSAs) are collected into a group and refreshed, checksummed, or aged, from 10 to 1800 seconds. The default is 240 seconds.
spf_delay	The delay time between when OSPF receives a topology change and when it starts a shortest path first (SPF) calculation in seconds, from 0 to 65535. The default is 5 seconds.
spf_holdtime	The hold time between two consecutive SPF calculations in seconds, from 0 to 65535. The default is 10 seconds.
stub	An OSPF area that carries a default route and intra- and inter-area routes but does not carry external routes. Virtual links cannot be configured across a stub area, and they cannot contain an autonomous system boundary router (ASBR).
subnets	(Optional) For redistributing routes into OSPF, scopes the redistribution for the specified protocol.
summary-address	Configures the summary address for OSPF redistribution.

tag_value	The value to match (for controlling redistribution with route maps).
timers	Configures timers for the OSPF process.

Defaults

The default is for OSPF routing to be disabled on the firewall.

The default value for *cost* is 1.

The default authentication type for an area is **0**, which means no authentication.

By default, OSPF routing through the firewall is compatible with RFC 1583.

The default for the **area** *area_id* **range** *ip_address netmask* [**advertise | not-advertise**] command is **advertise**.

The default for *d1*, *d2*, and *d3* in the **distance ospf** [**intra-area** *d1*][**inter-area** *d2*][**external** *d3*] subcommand is is 110.

By default, the **log-adj-changes** subcommand is enabled.

The default for *spf_delay* is 5 seconds, and the default for *spf_holdtime* is 10 seconds.

The default for the **timers lsa-group-pacing** *seconds* subcommand is 240 seconds.

No area is defined by default for the **area** *area_id* **nssa no-redistribution** or **area** *area_id* **default-information-originate** subcommands.

Command Modes

The **router ospf** command is available in configuration mode.

The **show router ospf** command is available in privileged mode.

Usage Guidelines

The **router ospf** command is the global configuration command for OSPF routing processes running on the firewall. This is the main command for all of the OSPF configuration commands.

Note Open Shortest Path First (OSPF) is used instead of Routing Information Protocol (RIP). Do not attempt to configure the firewall for both OSPF and RIP simeltaneously.

When using the **no** form of a **router ospf** command, optional arguments need not be specified unless they provide necessary information. The **no router ospf** command terminates the OSPF routing process specified by its *pid.*

PIX Firewall software Version 6.3 supports a maximum of two (2) OSPF processes.

The **show ospf** command displays the configured **router ospf** subcommands.

OSPF areas

The **area** *area_id* subcommand creates a regular OSPF area. The **no area** *area_id* command removes the OSPF area, whether it is regular, stubby, or not-so-stubby.

area *area_id* authentication message-digest

The default authentication type for an area is **0**, which means no authentication. To enable authentication for an OSPF area, use the **area** *area_id* **authentication message-digest** subcommand. To remove an authentication configuration from an area, use the **no area** *area_id* **authentication message-digest** subcommand.

area *area_id* **default-cost** *cost*

To specify a cost for the default summary route sent into a stub or not-so-stubby area (NSSA), use the **area** *area_id* **default-cost** *cost* subcommand. To remove the assigned default route cost, use the **no area** *area_id* **default-cost** subcommand. The default value for *cost* is 1.

area *area_id* **filter-list prefix** *prefix_list_name* [**in | out**]

To filter prefixes advertised in type 3 link-state advertisements (LSAs) between Open Shortest Path First (OSPF) areas of an Area Border Router (ABR), use the **area** *area_id* **filter-list prefix** *prefix_list_name* [**in | out**] subcommand. To change or cancel the filter, use the no **area** *area_id* **filter-list prefix** *prefix_list_name* [**in | out**] subcommand.

area *area_id* **nssa** [**no-redistribution**] [**default-information-originate** [**metric-type 1 | 2**] [**metric** *metric_value*]]

Routes that originate from other routing protocols (or different OSPF processes) and that are injected into OSPF through redistribution are called external routes. There are two forms of external metrics: type 1 and type 2. These routes are represented by `O E2` (for type 2) or `O E1` (for type 1) in the IP routing table, and they are examined after the firewall is done building its internal routing table. After they are examined, they are flooded throughout the autonomous systems (AS), unaltered. (Autonomous systems are a collection of networks, subdivided by areas, under a common administration sharing a common routing strategy.)

OSPF type 1 metrics result in routes adding the internal OSPF metric to the external route metric; they are also expressed in the same terms as an OSPF link-state metric. The internal OSPF metric is the total cost of reaching the external destination, including whatever internal OSPF network costs are incurred to get there. (These costs are calculated by the device wanting to reach the external route.) Because it is calculated this way, the OSPF type 1 metric is generally preferred.

OSPF type 2 metrics do not add the internal OSPF metric to the cost of external routes and are the default type used by OSPF. The use of OSPF type 2 metrics assumes that you are routing between autonomous systems (AS); therefore, the cost is considered greater than any internal metrics. This eliminates the need to add the internal OSPF metrics.

To configure an area as a not-so-stubby area (NSSA), use the **area** *area_id* **nssa** [**no-redistribution**] [**default-information-originate** [**metric-type 1 | 2**] [**metric** *metric_value*]] subcommand. To remove the entire NSSA configuration, use the **no area** *area_id* **nssa** subcommand. To remove a single NSSA configuration option, specify the option in the **no** subcommand. For example, to remove the **no-redistribution** option, use the **no area** *area_id* **nssa no-redistribution** command. By default, no NSSA is defined.

area *area_id* **range** *address netmask* [**advertise | not-advertise**]

To consolidate and summarize routes at an area boundary, use the **area** *area_id* **range** *address netmask* [**advertise | not-advertise**] subcommand. To disable this function, use the **no area** *area_id* **range** *ip_address netmask* subcommand. The **no area** *area_id* **range** *ip_address netmask* **not-advertise** subcommand removes only the **not-advertise** option.

area *area_id* **stub** [**no-summary**]

To define an area as a stub area, use the **area** *area_id* **stub** [**no-summary**] subcommand. To remove the stub area function, use the no **area** *area_id* **stub** [**no-summary**] subcommand. When **area** *area_id* **stub no-summary** is configured, you must use **no area** *area_id* **stub no-summary** to remove the no summary option. The default is for no stub areas to be defined.

[**no**] **area** *area_id* **virtual-link** *router_id* [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [**transmit-delay** *seconds*] [**dead-interval** *seconds*] [**authentication-key** *password*] [**message-digest-key** *id* **md5**] *password*

To define an OSPF virtual link, use the **area** *area_id* **virtual-link** *router-id* subcommand with the optional parameters. To remove a virtual link, use the **no area** *area_id* **virtual-link** *router_id* subcommand.

compatible rfc1583

To restore the method used to calculate summary route costs per RFC 1583, use the **compatible rfc1583** subcommand. To disable RFC 1583 compatibility, use the **no compatible rfc1583** subcommand.

By default, OSPF routing through the firewall is compatible with RFC 1583. The **compatible rfc1583** subcommand is displayed in the configuration only if disabled by the **no compatible rfc1583** subcommand, and then as "`no compatible rfc1583`".

distance ospf [intra-area *d1*][inter-area *d2*][external *d3*]

To define OSPF route administrative distances based on route type, use the **distance ospf** [**intra-area** *d1*][**inter-area** *d2*][**external** *d3*] subcommand. To restore the default value, use the **no distance ospf** subcommand. The default for *d1*, *d2*, and *d3* is 110.

ignore lsa mospf

To suppress the sending of syslog messages when the router receives link-state advertisement (LSA) for Type 6 Multicast OSPF (MOSPF) packets, which are unsupported, use the **ignore lsa mospf** subcommand. To restore the sending of these syslog messages, use the **no ignore lsa mospf** subcommand.

log-adj-changes

To configure the router to send a syslog message when an OSPF neighbor goes up or down, use the **log-adj-changes** [**detail**] subcommand. To turn off this function, use the **no log-adj-changes** subcommand. The **detail** option sends a syslog message for each state change, not just when a neighbor goes up or down.

By default, the **log-adj-changes** subcommand is enabled, but the **log-adj-changes** subcommand is only displayed in the OSPF configuration when the **detail** option is specified or when it has been disabled.

network *prefix ip_address netmask* area *area_id*

To define the interfaces on which OSPF runs and the area ID for those interfaces, use the **network** *prefix ip_address netmask* **area** *area_id* subcommand. To disable OSPF routing for the interfaces defined with the *prefix ip_address netmask* pair, use the **no network** *prefix ip_address netmask* **area** *area_id* subcommand.

summary-address *addr netmask*

To create aggregate addresses for OSPF, use the **summary-address** *addr netmask* [**not-advertise**] [**tag** *tag*] subcommand. To restore the default, use the no **summary-address** *addr netmask* subcommand. The *addr* value is the summary address designated for a range of addresses, and *netmask* is the IP subnet mask used for the summary route.

router-id

To use a fixed router ID, use the **router-id** *address* subcommand. To reset OSPF to use the previous OSPF router ID behavior, use the **no router-id** subcommand.

Note If the highest-level IP address on the firewall is a private address, then this address is sent in hello packets and database definitions (DBDs). To prevent this, set the **router-id** *ip_ddress* to a global address.

timers {**spf** *spf_delay spf_holdtime* | **lsa-group-pacing** *seconds*}

To configure the delay time between when OSPF receives a topology change and when it starts a shortest path first (SPF) calculation, and the hold time between two consecutive SPF calculations, use the **timers spf** *spf_delay spf_holdtime* subcommand. To return to the default timer values, use the **no timers spf** subcommand.

To change the interval at which OSPF link-state advertisements (LSAs) are collected into a group and refreshed, checksummed, or aged, use the **timers lsa-group-pacing** *seconds* subcommand. To restore the default value, use the no **timers lsa-group-pacing** *seconds* subcommand. The default for *seconds* is 240.

Examples

To enter subcommand mode on the outside interface of the firewall (needed to configure OSPF routing), enter the following command:

```
pixfirewall(config)# router ospf 5
pixfirewall(config-router)#
```

When in the routing subcommand mode, the command prompt appears as "**(config-router)#**".

Related Commands

prefix-list	Configures a prefix list to be used for OSPF routing.
route-map	Creates a route map for redistributing routes from one routing protocol to another. Used in configuring OSPF routing on the firewall.
routing interface	Configures interface-specific OSPF routing parameters.

routing interface

Configures interface-specific OSPF routing parameters. This command is the main command for all OSPF interface submode commands. (Use the **router ospf** command to configure global parameters and to enable OSPF routing through the firewall.) OSPF routing is not supported on the PIX 501.

[**no**] **routing interface** *interface_name*

Subcommands to the **routing interface** command:

[**no**] **ospf authentication** [**message-digest** | **null**]

[**no**] **ospf authentication-key** *password*

[**no**] **ospf cost** *interface_cost*

[**no**] **ospf database-filter all out**

[**no**] **ospf dead-interval** *seconds*

[**no**] **ospf hello-interval** *seconds*

[**no**] **ospf message-digest-key** *key-id* **md5** *key*

[**no**] **ospf mtu-ignore**

[**no**] **ospf priority** *number*

[**no**] **ospf retransmit-interval** *seconds*

[**no**] **ospf transmit-delay** *seconds*

Syntax Description

authentication-key *password*	Assigns an OSPF authentication password for use by neighboring routing devices. This can be any continuous string of keyboard characters, except for whitespace characters such as tabs or spaces, up to 8 bytes in length.
database-filter all out	Filters out outgoing link-state advertisements (LSAs) to an OSPF interface.
dead-interval *seconds*	Sets the interval before declaring a neighboring routing device is down if no hello packets are received, from 1 to 65535 seconds. This value must be the same for all nodes on the network. The default is four times the interval set by the **ospf hello-interval** command.
hello-interval *seconds*	Specifies the interval between hello packets sent on the interface, from 1 to 65535 seconds. The default is 10 seconds.
interface_cost	The cost (a link-state metric) of sending a packet through an interface. This is an unsigned integer value from 0 to 65535. **0** represents a network that is directly connected to the interface, and the higher the interface bandwidth, the lower the associated cost to send packets across that interface. In other words, a large cost value represents a low bandwidth interface and a small cost value represents a high bandwidth interface. The OSPF interface default cost on the firewall is **10**. This default differs from Cisco IOS software, where the default cost is **1** for fast Ethernet (FE) and Gigabit Ethernet (GE) and **10** for 10BaseT. This is important to take into account if you are using Equal Cost Multi-Path (ECMP) in your network.
interface_name	The name of the interface to configure.
key_id	A numerical ID number, from 1 to 255, for the authentication key.
md5 *key*	An alphanumeric password of up to 16 bytes. However, whitespaces characters such as a tab or space are not supported.
message-digest	Specifies to use OSPF message digest authentication.
message-digest-key	Enables Message Digest 5 (MD5) authentication. (MD5 verifies the integrity of the communication, authenticates the origin, and checks for timeliness.)
null	Specifies to not use OSPF authentication. This overrides password or message digest authentication (if configured) for an OSPF area.
ospf	Keyword for configuring interface-specific OSPF parameters.
priority *number*	A positive integer from 0 to 255 that specifies the priority of the router. The default is 1.
retransmit-interval *seconds*	Specifies the time between link-state advertisement (LSA) retransmissions for adjacent routers belonging to the interface, from 1 to 65535 seconds. The default is 5 seconds.
transmit-delay *seconds*	Sets the estimated time required to send a link-state update packet on the interface, from 1 to 65535 seconds. The default is 1 second.

Defaults

By default, OSPF routing is disabled on the firewall interfaces.

By default, the **mtu-ignore** subcommand is enabled.

The default value for the **ospf authentication** [**message-digest | null**] subcommand is **null**, which means no area authentication.

The default value for the **ospf dead-interval** subcommand is four times the interval set by the **ospf hello-interval** command.

The default value for the **ospf hello-interval** subcommand is 10 seconds.

The default value for the **ospf retransmit-interval** subcommand is 5 seconds.

The default value for the **ospf transmit-delay** subcommand is 1 second.

Command Modes

The **routing** command is available in configuration mode.

The **show routing** command is available in privileged mode.

Usage Guidelines

The **routing interface** *interface_name* command is the main command for all interface-specific OSPF interface mode commands. Enter this command with the name of the firewall interface (*interface_name*) that you want to configure, and then proceed with interface-specific configuration through the **routing interface** subcommands. You do not need to specify optional arguments in the **no** forms of the **routing interface** subcommands (unless they provide necessary information).

The **no routing interface** *interface_name* command removes the routing configuration for the interface specified only.

The **clear routing** command resets the interface-specific routing configuration to its defaults and removes the interface-specific routing configuration. However, this command does not remove any OSPF data structures that have been defined.

The **clear ospf** [*pid*] {**process | counters | neighbor** [**neighbor-intf**] [**neighbr-id**]} command resets the OSPF routing process ID, counters, neighbor interface router designation, or neighbor router ID, depending on the option selected. This command does not remove any configuration. Use the no form of the **router ospf** or **routing interface** command to remove the OSPF configuration.

The **show routing interface** *interface_name* command displays the configuration for the interface specified.

ospf authentication

To specify the authentication type for an interface, use the **ospf authentication** [**message-digest | null**] subcommand. To remove the authentication type for an interface, use the **no ospf authentication** [**message-digest | null**] subcommand. The default area authentication is **null**, which means no authentication.

ospf authentication-key

To assign a password to be used by neighboring routers that are using the OSPF simple password authentication, use the **ospf authentication-key** *password* subcommand. The variable *password* can be any continuous string of characters that can be entered from the keyboard, up to 8 bytes in length.

To remove a previously assigned OSPF password, use the **no ospf authentication-key** subcommand.

ospf cost

To explicitly specify the cost of sending a packet on an interface, use the **ospf cost** *interface_cost* subcommand. The *interface_cost* parameter is an unsigned integer value from 0 to 255, expressed as the link-state metric.

To reset the path cost to the default value, use the **no ospf cost** subcommand.

ospf database-filter all out

To filter outgoing link-state advertisements (LSAs) to an OSPF interface, use the **ospf database-filter** subcommand. To restore the forwarding of LSAs to the interface, use the **no ospf database-filter all out** subcommand.

ospf dead-interval

To set the dead interval before neighbors declare the router down (the length of time during which no hello packets are seen), use the **ospf dead-interval** *seconds* subcommand. The variable *seconds* specifies the dead interval and must be the same for all nodes on the network. The default for *seconds* is four times the interval set by the **ospf hello-interval** command, from 1 to 65535. To return to the default interval value, use the **no ospf dead-interval** subcommand.

ospf hello-interval

To specify the interval between hello packets that the firewall sends on the interface, use the **ospf hello-interval** *seconds* subcommand. To return to the default interval, use the **no ospf hello-interval** subcommand. The default is 10 seconds, with a range from 1 to 65535.

ospf mtu-ignore

The **ospf mtu-ignore** subcommand disables OSPF MTU mismatch detection on receiving DBD packets and is enbled by default.

ospf message-digest-key *key_id* md5 *key*

To enable OSPF Message Digest 5 (MD5) authentication, use the **ospf message-digest-key** *key_id* **md5** *key* subcommand. To remove an old MD5 key, use the **no ospf message-digest-key** *key_id* **md5** *key* subcommand. The *key_id* variable is a numerical identifier, from 1 to 255, for the authentication key, and the *key* variable is an alphanumeric password of up to 16 bytes.

ospf priority

To set the router priority, which helps determine the designated router for this network, use the **ospf priority** *number* subcommand. To return to the default value, use the **no ospf priority** *number* subcommand.

ospf retransmit-interval

To specify the time between link-state advertisement (LSA) retransmissions for adjacencies belonging to the interface, use the **ospf retransmit-interval** *seconds* subcommand. To return to the default value, use the **no ospf retransmit-interval** subcommand. The default value is 5 seconds, with a range from 1 to 65535.

ospf transmit-delay

To set the estimated time required to send a link-state update packet on the interface, use the **ospf transmit-delay** *seconds* subcommand. To return to the default value, use the **no ospf transmit-delay** subcommand. The default value is 1 second, with a range from 1 to 65535.

Examples

To enter subcommand mode on the outside interface of the firewall (needed to configure OSPF routing), enter the following command:

```
pixfirewall(config)# routing interface outside
pixdocipsec1(config-routing)#
```

When in the routing subcommand mode, the command prompt appears as "**(config-routing)#**".

The following example shows the configuration for two concurrently running OSPF processes, with the IDs 5 and 12, on the outside interface of the firewall:

```
pixfirewall(config)# routing interface
pixfirewall(config-routing)# show ospf

 Routing Process "ospf 5" with ID 127.0.0.1 and Domain ID 0.0.0.5
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x     0
 Number of opaque AS LSA 0. Checksum Sum 0x     0
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 0. 0 normal 0 stub 0 nssa
 External flood list length 0

 Routing Process "ospf 12" with ID 172.23.59.232 and Domain ID 0.0.0.12
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
 Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
 Number of external LSA 0. Checksum Sum 0x     0
 Number of opaque AS LSA 0. Checksum Sum 0x     0
 Number of DCbitless external and opaque AS LSA 0
 Number of DoNotAge external and opaque AS LSA 0
 Number of areas in this router is 0. 0 normal 0 stub 0 nssa
 External flood list length 0
```

The following example changes the retransmit interval to 15 seconds:

```
pixdocipsec1(config-routing)# ospf retransmit-interval 15
```

Related Commands

prefix-list	Configures a prefix list to be used for OSPF routing.
route-map	Creates a route map for redistributing routes from one routing protocol to another. Used in configuring OSPF routing on the firewall.
router ospf	Configures global parameters for the OSPF routing processes on the firewall, and enables or disables OSPF routing through the firewall.