



M through R Commands

mroute

Configures a static multicast route. (Configuration mode.)

Configure with the command...	Remove with the command...
<code>mroute src smask in-if-name dst dmask out-if-name</code>	<code>no mroute src smask in-if-name dst dmask out-if-name</code>

Show command options	Show command output
<code>show mroute [dst [src]]</code>	Displays the current multicast route table.

Syntax Description

<i>dmask</i>	The destination network address mask.
<i>dst</i>	The Class D address of the multicast group.
<i>in-if-name</i>	The input interface name to pass multicast traffic.
<i>out-if-name</i>	The output interface name to pass multicast traffic.
<i>smask</i>	The multicast source network address mask.
<i>src</i>	The IP address of the multicast source.

Usage Guidelines

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

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
```

multicast

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

Configure with the command...	Remove with the command...
multicast interface <i>interface_name</i> [max-groups <i>number</i>]	no multicast interface <i>interface_name</i> clear multicast
Subcommands to the multicast command:	Subcommands to the multicast command:
igmp forward interface <i>interface_name</i>	no igmp
igmp access-group <i>acl_id</i>	clear igmp [<i>group</i> interface <i>interface_name</i>]
igmp version {1 2}	
igmp join-group <i>group</i>	
igmp query-interval <i>seconds</i>	
igmp query-max-response-time <i>seconds</i>	

Show command options	Show command output
show igmp [<i>group</i> interface <i>interface_name</i>] [detail]	Displays the IGMP information for a multicast group, whether statically configured or dynamically created.
show multicast [interface <i>interface_name</i>]	Displays all or per-interface multicast settings. Also displays the IGMP configuration for any interface that is specified.

Syntax Description

<i>acl_id</i>	Access control list ID.
detail	Displays all information in the IGMP table.
<i>group</i>	The address of the multicast group.
igmp	Internet Group Management Protocol.
<i>interface_name</i>	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.
<i>number</i>	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.
<i>seconds</i>	Specifies the number of seconds to wait.

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.

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 an example of 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
```

mtu

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

Configure with the command...	Remove with the command...
mtu <i>if_name bytes</i>	no mtu [<i>if_name bytes</i>]

Show command options	Show command output
show mtu	Displays the current block size.

Syntax Description

<i>bytes</i>	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.
<i>if_name</i>	The internal or external network interface name.

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.

Examples

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

```
interface ethernet0 auto
mtu inside 8192

show mtu
mtu outside 1500
mtu inside 8192
```

name/names

Associate a name with an IP address. (Configuration mode.)

Configure with the command...	Remove with the command...
name <i>ip_address name</i>	no name [<i>ip_address name</i>] clear names
names	no names clear names
Show command options	Show command output
show names	Displays the name command statements in the configuration.

Syntax Description

<i>ip_address</i>	The IP address of the host being named.
<i>name</i>	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 <i>name</i> cannot start with a number. If the name is over 16 characters long, the name command fails.

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. (Configuration mode.)

Configure with the command...	Remove with the command...
<code>nameif hardware_id if_name security_level</code>	<code>clear nameif</code>
Show command options	Show command output
<code>show nameif</code>	Displays interface names.

Syntax Description

<i>hardware_id</i>	<p>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 <i>Cisco PIX Firewall Hardware Installation Guide</i>.</p> <p>A logical choice for an Ethernet interface is ethernetn. These names can also be abbreviated with any leading characters in the name, for example, ether1 or e2.</p>
<i>if_name</i>	<p>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 "intfn" where <i>n</i> is 2 through 5.</p>
<i>security_level</i>	<p>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).</p> <p>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.</p> <p>Interfaces with the same security level cannot communicate with each other. We recommend that every interface have a unique security level.</p>

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.

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 use of the **nameif** command:

```
nameif ethernet2 perimeter1 sec50
nameif ethernet3 perimeter2 sec20
```

Related Commands

- [interface](#)

nat

Associate a network with a pool of global IP addresses. (Configuration mode.)

Configure with the command...	Remove with the command...
nat [(if_name)] id address [netmask [outside] [dns] [norandomseq] [conn_limit [em_limit]]]	no nat [(if_name)] id address [netmask [outside]]
nat [(if_name)] 0 access-list acl_name	no nat [(if_name)] 0 access-list acl_name

Show command options	Show command output
show nat	Displays the nat command statements in the current configuration.

Syntax Description

access-list	Associates access-list command statements to the nat 0 command and exempts traffic that matches the access-list from NAT processing.
<i>acl_name</i>	The access list name.
clear nat	Removes nat command statements from the configuration.
<i>conn_limit</i>	The connection time limit.
dns	Specifies that DNS replies that match the xlate are translated.

<i>em_limit</i>	The embryonic connection limit. The default is 0, which means unlimited connections. Set it lower for slower systems, higher for faster systems.
<i>hh:mm:ss</i>	The timeout interval for the translation slot. However, timeout only occurs if no TCP or UDP connection is actively using the translation.
<i>id</i>	The id number to match with the global address pool.
<i>if_name</i>	The internal network interface name.
<i>local_ip</i>	Internal network IP address to be translated. You can use 0.0.0.0 to allow all hosts to start outbound connections. The 0.0.0.0 <i>local_ip</i> can be abbreviated as 0 .
<i>max_conns</i>	The maximum TCP connections permitted from the interface you specify.
<i>nat_id</i>	<i>nat_id</i> values can be 0 , 0 access list acl_name , or a number greater than zero (0). A <i>nat_id</i> that is 0 specifies the inside hosts for identity translation. Identity translations are translations that map an address to itself. The restriction is that the traffic must initiate from an inside host. A <i>nat_id</i> that is 0 access list acl_name specifies the traffic to exempt from NAT processing, based on the access list specified by <i>acl_name</i> . This is useful in Virtual Private Network (VPN) configuration where traffic between private networks should be exempted from NAT. A <i>nat_id</i> that is a number greater than zero (0) specifies the inside hosts for dynamic address translation. The dynamic addresses are chosen from a global address pool created with the global command, so the <i>nat_id</i> number must match the <i>global_id</i> number of the global address pool you want to use for dynamic address translation.
<i>netmask</i>	Network mask for <i>local_ip</i> . You can use 0.0.0.0 to allow all outbound connections to translate with IP addresses from the global pool. The netmask 0.0.0.0 can be abbreviated as 0 .
norandomseq	Do not randomize the TCP packet's sequence number. Only use this option if another inline firewall is also randomizing sequence numbers and the result is scrambling the data. Using this option disables TCP Initial Sequence Number (ISN) randomization protection. Without this protection, inside hosts with weak self-ISN protection become more vulnerable to TCP connection hijacking.
outside	Specifies that the nat command apply to the outside interface address. For access control, IPSec, and AAA use the real outside address.
timeout	Sets the idle timeout value for the translation slot.

Usage Guidelines

The **nat** command lets you enable or disable address translation for one or more internal addresses. Address translation means that when a host starts an outbound connection, the IP addresses in the internal network are translated into global addresses. Network Address Translation (NAT) allows your network to have any IP addressing scheme and the PIX Firewall protects these addresses from visibility on the external network.



Note

If not explicitly included in the **nat** command, 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.

The **nat** *if_name* **0** **access-list** *acl_name* command lets you exempt traffic that is matched by the **access-list** command statements from the NAT services. Adaptive Security remains in effect with the **nat 0 access-list** command. The extent to which the inside hosts are accessible from the outside depends on the **access-list** command statements that permit inbound access. The *if_name* is the higher security level interface name. The *acl_name* is the name you use to identify the **access-list** command statement.

With PIX Firewall software version 5.3 and higher, there is no longer a restriction on having the **nat 0** command (Identity NAT) and the **nat 0 access-list** command configured at the same time. Both the **nat 0** command and the **nat 0 access-list** command may be configured concurrently.

The **access-list** option changes the behavior of the **nat 0** command. (Without the **access-list** option, the command is backward compatible with previous versions.) The **nat 0** command implemented the identity feature; this new version of the command disables NAT. Specifically, the new behavior disables proxy ARPing for the IP addresses in the **nat 0** command statement.

**Note**

The access list you specify with the **nat 0 access-list** command will not work with an **access-list** command statement that contains a port specification. The following sample command statements will not work.

```
access-list no-nat permit tcp host xx.xx.xx.xx host yy.yy.yy.yy
nat (inside) 0 access-list no-nat
```

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

The connection limit lets you set the maximum number of outbound connections that can be started with the IP address criteria you specify. The embryonic connection limit lets you prevent a type of attack where processes are started without being completed. An embryonic connection is a connection that someone attempted but has not completed and has not yet seen data. Every connection is embryonic until it sets up.

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

The **nat outside** option lets you enable or disable outside NAT, which address translates the source address of a connection coming from a lower security interface to higher interface. This feature is also called Bi-Directional NAT. By default, address translation occurs only for host addresses on the higher security or "inside" interface.

**Note**

If outside dynamic NAT is enabled on an interface, explicit NAT policy must be configured for all hosts on the interface.

Use a *natid* of **0** with the **outside** option to disable address translation for host addresses on the lower security interface. Use this option only if outside dynamic NAT is configured on the interface. By default, address translation is automatically disabled for hosts connected to the lower security interface.

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

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.

Usage Notes

1. You can enable identity address translation with the **nat 0** command. Use this command when you have IP addresses that are the same as those used on more than one interface. Adaptive Security remains in effect with the **nat 0** command. The extent to which the inside hosts are accessible from the outside depends on the **access-list** command statements that permit inbound access.

Addresses on each interface must be on a different subnet. See Appendix D “TCP/IP Reference Information” of the *Cisco PIX Firewall and VPN Configuration Guide* for more information about subnetting.

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** command statements appear in the configuration.

2. The **nat 1 0 0** command means that all outbound connections can pass through the PIX Firewall with address translation. If you use the **nat (inside) 1 0 0** command, users can start connections on any interface with a lower security level, on the both perimeter interfaces and the outside interface. With NAT in effect, you must also use the **global** command statement to provide a pool of addresses through which translated connections pass. In effect, you use the **nat** command statement to specify from which interface connections can originate and you use the **global** command statement to determine at which interface connections can occur. The NAT ID must be the same on the **nat** and **global** command statements.
3. The **nat 1 10.2.3.0** command means that only outbound connections originating from the inside host 10.2.3.0 can pass through the PIX Firewall to go to their destinations with address translation.
4. The PIX Firewall does not support outside NAT for non-H.323 multimedia applications or between overlapping network addresses.

Examples

The **nat 0** command requires that traffic initiates from an inside host.

If you want the addresses to be visible from the outside network, 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 255.255.255.224
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 to permit internal host 10.1.1.15, accessible through the inside interface, “inside,” 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 will disable all NAT 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, and after all the 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 interface

```

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

```

Related Commands

- [global](#)
- [outbound/apply](#)

ntp

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

Configure with the command...	Remove with the command...
ntp authenticate	no ntp authenticate
ntp authentication-key <i>number md5 value</i>	no ntp authentication-key <i>number md5 value</i>
ntp server <i>ip_address</i> [key <i>number</i>] source <i>if_name</i> [prefer]	no ntp server <i>ip_address</i>
ntp trusted-key <i>number</i>	no ntp trusted-key <i>number</i>
N/A	clear ntp

Show command options	Show command output
show ntp	Displays the current NTP configuration.
show ntp associations [detail]	Displays the configured network time server associations.
show ntp status	Displays the NTP clock information.

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.
<i>if_name</i>	Specifies the interface to use to send packets to the network time server.
<i>ip_address</i>	The IP address of the network time server with which to synchronize.
key	Specifies the authentication key.
md5	The encryption algorithm.
<i>number</i>	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.
<i>value</i>	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.

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 **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.

Usage Notes

1. 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.
2. 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.
3. 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.

Examples

The following are examples of the **show ntp** commands.

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 (syncd), # master (unsyncd), + 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

- [clear](#)
- [debug](#)
- [show](#)

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. (Configuration mode.)

[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 netmask*

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

<i>begin_service</i>	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 <i>description_text</i>	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.
<i>end_service</i>	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.
<i>grp_id</i>	Required parameter that identifies the object group (one to 64 characters). Can be any combination of letters, digits, and the “_”, “-”, “.” characters.
<i>grp_type</i>	The type of group, either ICMP type, network, protocol, or service.
host	Keyword used with the <i>host_addr</i> parameter to define a host object.
<i>host_addr</i>	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.
<i>icmp_type</i>	The decimal number or name of an ICMP type.
<i>net_addr</i>	The network address. Used with <i>netmask</i> to define a subnet object.
<i>netmask</i>	The netmask. Used with <i>net_addr</i> 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.
<i>obj_grp_id</i>	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.
<i>protocol</i>	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.

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_name permit tcp any object-group group_name
```

Additionally, the **access-list** and **conduit** command parameters can be grouped as follows in [Table 7-2](#).

Table 7-2 Object Groups to Replace Individual Parameters

Instead of using individual parameters...	...use the following object group:
<i>protocol</i>	object-group protocol
<i>host and subnet</i>	object-group network
<i>service</i>	object-group service
<i>icmp_type</i>	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 **access-list** to close an **object-group** subcommand mode and exit the **object-group** main command.

Use 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 **show object-group** 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 are not being used in a command for that group type only.

For use in the **object-group icmp-type** command, [Table 7-3](#) lists ICMP type numbers and names:

Table 7-3 ICMP 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 concatenated 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:

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

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

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

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

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

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

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

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

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

```
(config-service)#exit
```

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

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

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

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

(config)# access-list grp_1 permit tcp object-group host_grp_1 any eq ftp
(config)# access-list grp_2 permit tcp object-group host_grp_2 any eq smtp
(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.pyl.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. (Configuration mode.)

Configure with the command...	Remove with the command...
apply [(if_name)] list_ID outgoing_src outgoing_dest	no apply [(if_name)] list_ID outgoing_src outgoing_dest clear apply
outbound list_ID permit deny ip_address [netmask [port[-port]] [protocol]	no outbound [list_ID permit deny ip_address [netmask [port[-port]] [protocol]] clear outbound
outbound list_ID except ip_address [netmask [port[-port]] [protocol]	no outbound [list_ID except ip_address [netmask [port[-port]] [protocol]]
Show command options	Show command output
show apply [(if_name)] [list_ID outgoing_src outgoing_dest]	Displays the apply command statements in the configuration.
show outbound	Displays the outbound command statements in the configuration.

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.
<i>if_name</i>	The network interface originating the connection.
<i>ip_address</i>	The IP address for this access list entry. Do not specify a range of addresses. The 0.0.0.0 <i>ip_address</i> can be abbreviated as 0.

<i>list_ID</i>	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. <i>List_IDs</i> are processed sequentially in descending order. For more information, see “ Outbound List Rules .”
<i>netmask</i>	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 <i>netmask</i> 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.
<i>port</i>	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.
<i>protocol</i>	Limit outbound access to udp , tcp , or icmp protocols. If a protocol is not specified, the default is tcp .

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_IDs* 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.

- 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).

- 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.
- The maximum number of **outbound** list entries in a configuration is 1599.
- Outbound lists have no effect on **access-list** command statement groups.
- 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). 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. (Privileged mode.)

Set with the command...	Remove with the command...
<code>pager [lines number]</code>	<code>clear pager</code>
	<code>no pager</code>

Show command options	Show command output
<code>show pager</code>	Displays pager status.

Syntax Description

<i>number</i>	The number of lines before the “---more---” prompt appears. The minimum is 1 . Use 0 to disable paging.
---------------	---

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 --->
```

passwd

Set password for Telnet access to the PIX Firewall console. (Privileged mode.)

Set with the command..	Remove with the command..
<code>passwd <i>password</i> [encrypted]</code>	<code>clear passwd</code>

Show command options	Show command output
<code>show passwd</code>	Displays the Telnet password.

Syntax Description

encrypted	Specifies that the password you entered is already encrypted. The <i>password</i> you specify with the encrypted option must be 16 characters in length.
<i>password</i>	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.

Usage Guidelines

The **passwd** command sets a password for Telnet access to the PIX Firewall console. An empty password is also 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 passwd** command resets the password to “cisco.”

**Note**

Write down the new password and store it in a manner consistent with your site’s security policy. Once you change this password, you cannot view it again.

Examples

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

```
passwd watag00s1am
show passwd
passwd jMorNbK0514fadBh encrypted
```

Related Commands

- [enable](#)

pdm

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

Display with the command...	Remove with the command...
<code>show pdm sessions</code>	<code>pdm disconnect <i>session_id</i></code> <code>clear pdm</code>
<code>pdm history enable</code>	<code>no pdm history enable</code>
<code>pdm history [view {all 12h 5d 60m 10m}] [snapshot] [feature {all blocks cpu failover ids interface <i>if_name</i> memory perfmon xlates}] [pdmclient]</code>	<code>no pdm history [view {all 12h 5d 60m 10m}] [snapshot] [feature {all blocks cpu failover ids interface <i>if_name</i> memory perfmon xlates}] [pdmclient]</code>
<code>pdm location <i>ip_address netmask if_name</i></code>	<code>clear pdm</code>
<code>pdm logging [level [<i>messages</i>]]</code>	<code>no pdm logging</code>

Show command options	Show command output
<code>show pdm history</code>	Displays the contents of the PDM history buffer.
<code>show pdm logging</code>	Displays the contents of the PDM buffer within PDM.
<code>show pdm sessions</code>	Displays a <i>session_id</i> for each active PDM session to the PIX Firewall, beginning with session number 0.

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.
blocks	History for system buffers. Similar to output of 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 of the show cpu usage command.
failover	History for failover. Similar to output of 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.
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).
<i>if_name</i>	Specifies the interface name on which PDM resides.
<i>ip_address</i>	Specifies the host or network on which PDM resides.
<i>level</i>	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.
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 of the show memory command.
<i>messages</i>	Specifies the number of messages stored in the PDM buffer. Once the buffer is full, old messages will be discarded.
<i>netmask</i>	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 of show perfmon command.
<i>session_id</i>	PDM session ID number available from the show pdm sessions command.
snapshot	Displays only the last PDM history data point.
xlates	History for translation slot information. Similar to output of the show xlate command.

Defaults

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

Usage Guidelines

The **pdm disconnect** command and the **show pdm sessions** command are accessible through the command line. The **clear pdm**, **pdm history** commands, **pdm location**, and **pdm logging** commands may appear in your configuration and are available through the CLI, but they are designed to work as internal PDM-to-PIX Firewall commands accessible through PDM.

The **pdm disconnect** command lets you disconnect a specific PDM session using a *session_id* obtained with the **show pdm sessions** command. The **show pdm sessions** command lists all the open PDM sessions going to a PIX Firewall.

The **pdm location** command can only associate one interface to an *ip_address /netmask* pair. Specifying an existing pair will replace the old definition. The PDM syslog messages are stored separately from the PIX Firewall syslog accessed through the **logging buffered** command.

The **clear pdm location** command will remove all of the PDM locations. 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 **clear pdm logging** command will clear the PDM log without disabling it.

Examples

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:PIXoutsideINTERF
ACE: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|M
ETRIC_HISTORY|SNAP|OBR|VIEW|10|64|METRIC_HISTORY|SNAP|IPR|VIEW|10|0|METRIC_HISTORY|SNAP|OP
R|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_HIST
ORY|SNAP|BLK80|VIEW|10|400|METRIC_HISTORY|SNAP|BLK256|VIEW|10|998|METRIC_HISTORY|SNAP|BLK1
550|VIEW|10|676|METRIC_HISTORY|SNAP|XLATES|VIEW|10|0|METRIC_HISTORY|SNAP|CONNS|VIEW|10|0|M
ETRIC_HISTORY|SNAP|TCPCONNS|VIEW|10|0|METRIC_HISTORY|SNAP|UDPCONNS|VIEW|10|0|METRIC_HISTOR
Y|SNAP|URLS|VIEW|10|0|METRIC_HISTORY|SNAP|WEBSNS|VIEW|10|0|METRIC_HISTORY|SNAP|TCPFIXUPS|V
IEW|10|0|METRIC_HISTORY|SNAP|TCPINTERCEPTS|VIEW|10|0|METRIC_HISTORY|SNAP|HTTPFIXUPS|VIEW|1
0|0|METRIC_HISTORY|SNAP|FTPFIXUPS|VIEW|10|0|METRIC_HISTORY|SNAP|AAAAUTHENUPS|VIEW|10|0|MET
RIC_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
rexid Portmap Request: [ 10s] : 0
rexid 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 Authorization 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

- [copy](#)
- [http](#)
- [setup](#)

perfmon

View performance information. (Privileged mode.)

Display with the command...	Remove with the command...
perfmon verbose	perfmon quiet
perfmon interval <i>seconds</i>	
perfmon settings	N/A

Show command options	Show command output
show perfmon	Displays PIX Firewall performance information. (However, this command output does not display in a Telnet console session.)

Syntax Description

interval <i>seconds</i>	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.

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.

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. (Privileged mode.)

Test with the command...	Remove with the command...
ping [<i>if_name</i>] <i>ip_address</i>	N/A

Syntax Description

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

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
```

privilege

Configures or displays command privilege levels. (Configuration mode.)

Set with the command...	Remove with the command...
privilege [show clear configure] level level [mode enable configure] command command	no privilege [show clear config] level level [mode enable configure] command command
Show command options	Show command output
show curpriv	Displays the current privileges for a user.
show privilege [all command command level level]	Displays the privileges for a command or set of commands.

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.)
<i>command</i>	The command on which to set the privilege level.
configure	Sets the privilege level for the configure command corresponding to the command specified.
<i>configure</i>	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.
<i>enable</i>	For commands with both enable and configure modes, this indicates that the level is for the enable mode of the command.
<i>level</i>	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.

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.

Examples

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

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

Also, 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](#)
- [login](#)
- [object-group](#)
- [username](#)

quit

Exit configuration or privileged mode. (All modes.)

Exit with the command...	Access with the command...
quit	login

Syntax Description

quit	Exits the current privilege level or mode.
-------------	--

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. (Privileged mode.)

Reset with the command...	Remove with the command...
reload	N/A
reload noconfirm	

Syntax Description

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

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 RIP settings. (Configuration mode.)

Configure with the command...	Remove with the command...
rip <i>if_name</i> default passive [version [1 2]] [authentication [text md5 <i>key</i> (<i>key_id</i>)]]	no rip <i>if_name</i> default passive [version [1 2]] [authentication [text md5 <i>key</i> (<i>key_id</i>)]]
debug rip [<i>if_name</i>]	clear rip N/A

Show command options	Show command output
show rip [<i>if_name</i>]	Displays the current RIP settings.

Syntax Description

authentication	Enable RIP version 2 authentication.
default	Broadcast a default route on the interface.
<i>if_name</i>	The internal or external network interface name.
<i>key</i>	Key to encrypt RIP updates. This value must be the same on the routers and any other device <i>that provides RIP version 2 updates</i> . The <i>key</i> is a text string of up to 16 characters in length.
<i>key_id</i>	Key identification value. The <i>key_id</i> can be a number from 1 to 255. Use the same <i>key_id</i> 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.

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. (Configuration mode.)

Configure with the command...	Remove with the command...
route <i>if_name ip_address netmask gateway_ip [metric]</i>	clear route [<i>if_name ip_address [netmask gateway_ip]</i>] no route [<i>if_name ip_address [netmask gateway_ip]</i>]

Show command options	Show command output
show route	Displays the routes in the configuration.

Syntax Description

<i>gateway_ip</i>	Specify the IP address of the gateway router (the next hop address for this route).
<i>if_name</i>	The internal or external network interface name.
<i>ip_address</i>	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 .
<i>metric</i>	Specify the number of hops to <i>gateway_ip</i> . 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.
<i>netmask</i>	Specify a network mask to apply to <i>ip_address</i> . Use 0.0.0.0 to specify a default route. The 0.0.0.0 netmask can be abbreviated as 0 .

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.

If the **route** command statement uses the IP address from one of the PIX Firewall unit’s interfaces 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
```

