



Controlling Network Access and Use

This chapter describes how to establish and control network connectivity for different applications and implementations after you have completed your basic configuration, described in [Chapter 2](#), “Establishing Connectivity.” This chapter contains the following sections:

- [Enabling Server Access with Static NAT, page 3-1](#)
- [Enabling Inbound Connections, page 3-2](#)
- [Controlling Outbound Connectivity, page 3-4](#)
- [Using the Static Command for Port Redirection, page 3-5](#)
- [Using Authentication and Authorization, page 3-8](#)
- [Access Control Configuration Example, page 3-14](#)
- [Using TurboACL, page 3-18](#)
- [Downloading Access Lists, page 3-20](#)
- [Simplifying Access Control with Object Grouping, page 3-24](#)
- [Filtering Outbound Connections, page 3-31](#)

Enabling Server Access with Static NAT

Static Network Address Translation (NAT) creates a permanent, one-to-one mapping between an address on an internal network (a higher security level interface) and a perimeter or external network (lower security level interface). For example, to share a web server on a perimeter interface with users on the public Internet, use static address translation to map the server’s actual address to a registered IP address. Static address translation hides the actual address of the server from users on the less secure interface, making casual access by unauthorized users less likely. Unlike NAT or PAT, it requires a dedicated address on the outside network for each host, so it does not save registered IP addresses.

If you use a **static** command to allow inbound connections to a fixed IP address, use the **access-list** and **access-group** commands to create an access list and to bind it to the appropriate interface. For more information, refer to “[Enabling Inbound Connections](#).”



Note

Do not use the PIX Firewall interface address with the **static** command if Stateful Failover is enabled. Doing this will prevent Stateful Failover from receiving its interface monitoring probes, which run over IP protocol 105, and as a result, the interface will appear to be in a waiting state. For further information about Stateful Failover, refer to [Chapter 10](#), “Using PIX Firewall Failover.”

The main options of the **static** command are as follows:

```
static [(internal_if_name, external_if_name)] global_ip local_ip [netmask network_mask]
[max_conns]
```

- Replace *internal_if_name* with the internal network interface name. In general, this is the higher security level interface you are accessing.
- Replace *external_if_name* with the external network interface name. In general, this is the lower security level interface you are accessing.
- Replace *global_ip* with the outside (global) IP address. In general, this is the interface with the lower security level. This address cannot be a PAT IP address.
- Replace *local_ip* with the internal (local) IP address from the inside network. In general, this is the interface with the higher security level.
- Replace *network_mask* with the network mask that pertains to both *global_ip* and *local_ip*. For host addresses, always use 255.255.255.255. For network addresses, use the appropriate subnet mask for the network.
- (Optional) replace *max_conns* with the maximum number of concurrent connections permitted through the static address translation.



Note To configure static translation for a host residing on the less secure interface (using outside NAT) reverse the interface in the **static** command. Refer to the *Cisco PIX Firewall Command Reference* for more information about the **static** command.

For example, the following command maps a server with an internal IP address of 10.1.1.3 to the registered IP address 209.165.201.12:

```
static (inside, outside) 209.165.201.12 10.1.1.3 netmask 255.255.255.255
```

This command simply maps the addresses; make sure you also configure access using the **access-list** and **access-group** commands, as described in the next section. Also, you must inform the DNS administrator to create an MX record for the external address so that traffic sent to the server host name is directed to the correct address.



Note For more information about how to configure static translation without NAT, refer to the **static** command in the *Cisco PIX Firewall Command Reference*.

Enabling Inbound Connections

By default, the PIX Firewall denies access to an internal or perimeter (more secure) network from an external (less secure) network. You specifically allow inbound connections by using access lists. Access lists work on a first-match basis, so for inbound access, you must deny first and then permit after.



Note Beginning with Version 5.3, the PIX Firewall uses access lists to control connections between inside and outside networks. Access lists are implemented with the **access-list** and **access-group** commands. These commands are used instead of the **conduit** and **outbound** commands, which were used in earlier versions of PIX Firewall. In PIX Firewall software releases later than Version 6.3, the **conduit** and **outbound** commands are no longer supported. To help you with the conversion process, a tool is available online at: <https://cco-dev.cisco.com/cgi-bin/Support/OutputInterpreter/home.pl>.

You use the **access-list** and **access-group** commands to permit access based on source or destination IP address, or by the protocol port number. Use the **access-list** command to create a single access list entry, and use the **access-group** command to bind one or more access list entries to a specific interface. Only specify one **access-group** command for each interface.

**Note**

To allow access only for specific users, set up authentication, as described in “[Using Authentication and Authorization](#).”

Before you can set up an access list for a host, set up address translation by using a **global** or **static** command. Setting up address translation with the **global** command is described in [Chapter 2](#), “[Establishing Connectivity](#).” Setting up address translation using the **static** command was described earlier in the previous section “[Enabling Server Access with Static NAT](#).”

The **access-list** command has many features, some of which are described in the following sections:

- [Using TurboACL, page 3-18](#)
- [Downloading Access Lists, page 3-20](#)
- [Simplifying Access Control with Object Grouping, page 3-24](#)

For the complete syntax of the **access-list** command, see the *Cisco PIX Firewall Command Reference*.

The basic syntax for the **access-list** command is as follows:

```
access-list ID [line line-num] {deny|permit} protocol <source_address | interface if_name>
[operator port] destination_address [operator port]
```

- Replace *ID* with a name or number you create to identify a group of **access-list** command statements; for example, “acl_inbound,” which identifies that the permissions apply to access from the outside interface.
- To insert a remark or an access control entry (ACE), use the **line** keyword. Replace *line-num* with the line number at which to make the insertion.
- Use **permit** or **deny** depending on whether you want to permit or deny access to the server. By default, all inbound access is denied, so you must permit access to a specific protocol or port.
- Replace *protocol* with the protocol (tcp or udp). For most servers, such as HTTP or email, use **tcp**. For a complete list of permitted keywords and well-known port assignments, see “[Protocols and Applications](#)” in [Appendix D](#), “[TCP/IP Reference Information](#).”
- Replace *source_address* with the host or network address for those systems on the lower security level interface that must access the *destination_address*. Use **any** to let any host access the *destination_address*. If you specify a single host, precede the address with **host**; for example **host 192.168.1.2**. If you specify a network address, also specify a network mask; for example, **192.168.1.0 255.255.255.0**. Do not use the **host** keyword with network addresses.

Use the **interface** keyword if the interface has a dynamically assigned IP address. Replace *if_name* with the name of the interface configured using the **nameif** command.

- Use an *operator* to match port numbers used by the source or destination. The permitted operators are as follows:
 - lt—less than
 - gt—greater than
 - eq—equal to
 - negq—not equal to
 - range—an inclusive range of values

- Use the first *port* parameter after an operator to identify the protocol port used by the source host that initiates the connection.
- Replace *destination_address* with the host or network global address that you specified with the **static** command statement. For a host address, precede the address with **host**; for networks, specify the network address and the appropriate network mask.
- Use the second *port* parameter after an operator to specify the protocol port used by the destination host. For example, to identify a web server, use **eq http** or **eq 80**. For an email server, use **eq smtp** or **eq 25**. For a complete list of permitted keywords and well-known port assignments, see “Ports” in [Appendix D, “TCP/IP Reference Information.”](#)

Two **access-list** command statement definitions are required to permit access to the following ports:

- DNS, Discard, Echo, Ident, NTP, RPC, SUNRPC, and Talk each require one definition for TCP and one for UDP.
- TACACS+ requires one definition for port 49 on TCP.

The format for the **access-group** command is as follows:

```
access-group ID in interface low_interface
```

Replace *ID* with the same identifier that you specified in the **access-list** command statement.

Replace *low_interface* with the lower security interface that you specified in the **static** command statement. This is the interface through which users will access the external (global) address.

The following example illustrates the three commands required to enable access to a web server with the external IP address 209.165.201.12:

```
static (inside, outside) 209.165.201.12 10.1.1.3 netmask 255.255.255.255 0 0
access-list acl_out permit tcp any host 209.165.201.12 eq www
access-group acl_out in interface outside
```

This example uses the same **static** command that was shown in the previous section.

Controlling Outbound Connectivity

By default, all connections initiated on a network with a higher security level are allowed out, and you configure any restrictions required. You can control outbound access by IP address and protocol port, or combine access control with user authentication, as described in “[Using Authentication and Authorization.](#)” If you are not enforcing restrictions on outbound network traffic, outbound access lists are not required.

An outbound access list lets you restrict hosts from starting outbound connections or lets you restrict hosts from accessing specific destination addresses or networks. Access lists work on a first-match basis, so for outbound access lists, you must permit first and then deny after.

For example, you could restrict some hosts from accessing web sites or permit others access. Define access restrictions with the **access-list** command, and use the **access-group** command to bind the **access-list** command statements to an interface.

When creating an outbound access list, the basic syntax for the **access-list** command statement is the same as shown earlier in “[Enabling Inbound Connections:](#)”

```
access-list ID {deny|permit} protocol source_address [operator port] destination_address
[operator port]
```

Use the **deny** parameter to restrict specific types of access. For example, to prevent hosts belonging to the 192.168.1.0 network on the inside interface from starting connections on the outside interface and to permit all others, specify the 192.168.1.0 network address as the source address and the network connected to the outside interface as the destination address. In the example that follows, the network on the outside interface is 209.165.201.0. The **access-list** and **access-group** command statements are as follows.

```
access-list acl_in deny tcp 192.168.1.0 255.255.255.224 209.165.201.0 255.255.255.224
access-list acl_in permit ip any any
access-group acl_in in interface inside
```

You can also use access lists to prevent access to a specific server. For example, if you want to restrict hosts on the inside interface from accessing a website at address 209.165.201.29 on the outside interface, use the following commands.

```
access-list acl_in deny tcp any host 209.165.201.29 eq www
access-list acl_in permit ip any any
access-group acl_in in interface inside
```

These commands let any hosts start connections, but not to 209.165.201.29. The **access-group** command specifies that the hosts are on the inside interface.

**Note**

You can use URL filtering for greater control of outbound access to web sites, as described in the “[Filtering URLs with Internet Filtering Servers](#)” section on page 3-32.”

Using the Static Command for Port Redirection

This section describes the port redirection feature, introduced in PIX Firewall Version 6.0. It includes the following topics:

- [Overview, page 3-5](#)
- [Port Redirection Configuration, page 3-6](#)
- [Port Redirection Example, page 3-7](#)

Overview

Port redirection allows hosts on a lower security interface to connect to a particular IP address and port and to have the PIX Firewall redirect the traffic to the appropriate server on a higher security interface.

The shared address can be a unique address, a shared outbound PAT address, or an address shared with the external interface. To implement port redirection, use the following command.

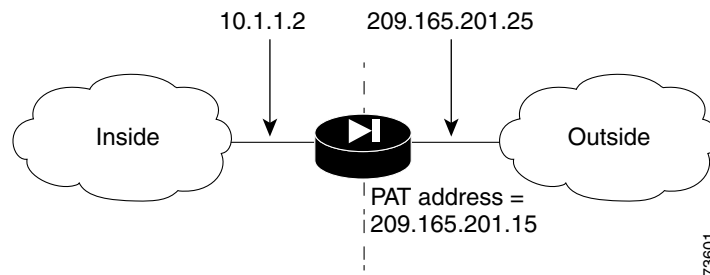
```
static [(internal_if_name, external_if_name)] {tcp|udp} {global_ip|interface} global_port
local_ip local_port [netmask mask]
```

For an explanation of this command syntax, refer to the *Cisco PIX Firewall Command Reference*.

Port Redirection Configuration

Figure 3-1 illustrates a typical network scenario in which the port redirection feature might be useful.

Figure 3-1 Port Redirection Using the Static Command



In the configuration described in this section, port redirection occurs for hosts on external networks as follows:

- Telnet requests to unique IP address 209.165.201.5 are redirected to 10.1.1.6
- FTP requests to unique IP address 209.165.201.5 are redirected to 10.1.1.3
- Telnet requests to PAT address 209.165.201.15 are redirected to 10.1.1.4
- Telnet requests to the PIX Firewall outside IP address 209.165.201.25 are redirected to 10.1.1.5
- HTTP request to PIX Firewall outside IP address 209.165.201.25 are redirected to 10.1.1.5
- HTTP port 8080 requests to PAT address 209.165.201.15 are redirected to 10.1.1.7 port 80

To implement this scenario, complete the following steps:

-
- Step 1** Configure application inspection of FTP requests on port 21 by entering the following command:
- ```
fixup protocol ftp 21
```
- Step 2** Configure the IP address of the lower and higher security interfaces of your PIX Firewall by entering the following command:

```
ip address outside 209.165.201.25 255.255.255.0
ip address inside 10.1.1.2 255.255.255.0
```

**Step 3** Identify a global PAT address for the lower security interface by entering the following command:

```
global (outside) 1 209.165.201.15
```

**Step 4** Configure NAT and PAT by entering the following command:

```
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
```

**Step 5** Redirect Telnet requests for 209.165.201.5:

```
static (inside,outside) tcp 209.165.201.5 telnet 10.1.1.6 telnet netmask 255.255.255.255 0
0
```

This command causes Telnet requests to be redirected to 10.1.1.6.

**Step 6** Redirect FTP requests for IP address 209.165.201.5:

```
static (inside,outside) tcp 209.165.201.5 ftp 10.1.1.3 ftp netmask 255.255.255.255 0 0
```

This command causes FTP requests to be redirected to 10.1.1.3.

**Step 7** Redirect Telnet requests for PAT address 209.165.201.15:

```
static (inside,outside) tcp 209.165.201.15 telnet 10.1.1.4 telnet netmask 255.255.255.255 0 0
```

This command causes Telnet requests to be redirected to 10.1.1.4.

**Step 8** Redirect Telnet requests for the PIX Firewall outside interface address:

```
static (inside,outside) tcp interface telnet 10.1.1.5 telnet netmask 255.255.255.255 0 0
```

This command causes Telnet requests to be redirected to 10.1.1.5.

**Step 9** Redirect HTTP requests for the PIX Firewall outside interface address:

```
static (inside,outside) tcp interface www 10.1.1.5 www netmask 255.255.255.255 0 0
```

This command causes HTTP request to be redirected to 10.1.1.5.

**Step 10** Redirect HTTP requests on port 8080 for PAT address 209.165.201.15:

```
static (inside,outside) tcp 209.165.201.15 8080 10.1.1.7 www netmask 255.255.255.255 0 0
```

This command causes HTTP port 8080 requests to be redirected to 10.1.1.7 port 80.

## Port Redirection Example

[Example 3-1](#) illustrates the configuration required to implement the port redirection described in this scenario.

### **Example 3-1** Port Redirection with the static Command

```
fixup protocol ftp 21
ip address outside 209.165.201.25 255.255.255.0
ip address inside 10.1.1.2 255.255.255.0
global (outside) 1 209.165.201.15
nat (inside) 1 0.0.0.0 0.0.0.0 0 0
static (inside,outside) tcp 209.165.201.5 telnet 10.1.1.6 telnet netmask 255.255.255.255 0 0
static (inside,outside) tcp 209.165.201.5 ftp 10.1.1.3 ftp netmask 255.255.255.255 0 0
static (inside,outside) tcp 209.165.201.15 telnet 10.1.1.4 telnet netmask 255.255.255.255 0 0
static (inside,outside) tcp interface telnet 10.1.1.5 telnet netmask 255.255.255.255 0 0
static (inside,outside) tcp interface www 10.1.1.5 www netmask 255.255.255.255 0 0
static (inside,outside) tcp 209.165.201.15 8080 10.1.1.7 www netmask 255.255.255.255 0 0
```

# Using Authentication and Authorization

You can use access lists to control traffic based on IP address and protocol, but to control access and use for specific users or groups, you must use authentication and authorization. Authentication, which is the process of identifying users, is supported by the PIX Firewall for RADIUS and TACACS+ servers. Authorization identifies the specific permissions for a given user.

If you want to apply authentication and authorization when an internal (local) host initiates a connection to an external (lower security) network, enable it on the internal (higher security) interface. To set up authentication and authorization to occur when an external host initiates a connection to an internal host, enable it on the outside interface.

**Note**

If you want a host on an outside (lower security level) interface to initiate connections with a host on an internal (higher security level) interface, create **static** and **access-list** command statements for the connection.

This section includes the following topics:

- [Configuring AAA, page 3-8](#)
- [Enabling Secure Authentication of Web Clients, page 3-10](#)
- [Configuring RADIUS Authorization, page 3-12](#)
- [Using MAC-Based AAA Exemption, page 3-13](#)

## Configuring AAA

To enable authentication and authorization, you must complete the following:

- Identify the IP address of the authentication server that you will use and determine a server encryption key to be shared by the authentication server and the PIX Firewall.
- Configure the authentication server with the users that can access the network, the services that they can use, and the hosts that they can access.
- Configure the PIX Firewall to either enable or disable authentication or authorization.

In addition, you can configure the PIX Firewall to control user access to specific hosts or services. However, it is easier to maintain this kind of access control in a single location, at the authentication server. After you enable authentication and authorization, the PIX Firewall prompts users of FTP, Telnet, or HTTP (Web) access. Controlling access to a specific system or service is handled by the authentication and authorization server.

**Note**

When using PIX Firewall Version 6.3 or higher, you can enable authentication with a user database that you configure locally on your PIX Firewall. The configuration steps are similar to those for configuring a RADIUS/TACACS+ server. The differences are noted within each step in the following procedure. For information about configuring the PIX Firewall local user database, refer to [“User Authentication”](#) in [Chapter 3, “Controlling Network Access and Use.”](#)

Follow these steps to enable the PIX Firewall to support user authentication and authorization:

- Step 1** For inbound authentication, create the **static** and **access-list** command statements required to permit outside hosts to access servers on the inside network.
- Step 2** If the internal network connects to the Internet, create a global address pool of registered IP addresses. Then specify the inside hosts that can start outbound connections with the **nat** command using the **access-list** command.
- Step 3** Identify the server that handles authentication or authorization using the **aaa-server** command. Create a unique server group name.

For example:

```
aaa-server AuthInbound protocol tacacs+
aaa-server AuthInbound (inside) host 10.1.1.1 TheUauthKey
aaa-server AuthOutbound protocol tacacs+
aaa-server AuthOutbound (inside) host 10.1.1.2 TheUauthKey
```



**Note** This step is not required when using the LOCAL database for authentication.

The first command statement creates the AuthInbound authentication group using TACACS+ authentication. The second command statement states that the AuthInbound server is on the inside interface, that its IP address is 10.1.1.1, and the encryption key is “TheUauthKey.”

The third command statement creates the AuthOutbound authentication group using TACACS+ authentication. The fourth command statement states that the AuthOutbound server is on the inside interface, that its IP address is 10.1.1.2, and the encryption key is “TheUauthKey.”



**Note** RADIUS authorization is provided with the **access-list** command statement as described in [“Configuring RADIUS Authorization.”](#)

- Step 4** Enable authentication with the **aaa authentication** command:

```
aaa authentication include authen_service if_name 0 0 0 0 <server_tag|LOCAL>
```

Replace *authen\_service* with an identifier that specifies the traffic to be included, such as **ftp**, **telnet**, **http** or **https**. For details about this option, refer to the **aaa authentication** command in the *Cisco PIX Firewall Command Reference*.

Replace *if\_name* with the name of the interface on which you are enabling authentication, as configured with the **nameif** command. To use the LOCAL database for authentication use the LOCAL keyword. To use a AAA server, replace *server\_tag* with the AAA server group name defined with the **aaa-server** command. For example:

```
aaa authentication include ftp outside 0 0 0 0 AuthOutbound
aaa authentication include telnet outside 0 0 0 0 AuthOutbound
aaa authentication include http outside 0 0 0 0 AuthOutbound
aaa authentication include ftp inside 0 0 0 0 AuthInbound
aaa authentication include telnet inside 0 0 0 0 AuthInbound
aaa authentication include http inside 0 0 0 0 AuthInbound
```



**Note** Be careful to apply authentication only to protocols that can be authenticated. Applying authentication using the **any** keyword will prevent protocols such as SMTP from passing through the PIX Firewall.

- Step 5** Enable authorization with the **aaa authorization** command. PIX Firewall checks the authorization request with the AAA server, which makes the decision about what services a user can access.

```
aaa authorization include authen_service if_name 0 0 0 0
```

Replace *authen\_service* with an identifier that specifies the traffic to be included, such as **ftp**, **telnet**, or **http**.




---

**Note** This step is not required when using the LOCAL database for authentication.

---

For example:

```
aaa authorization include ftp outside 0 0 0 0
aaa authorization include telnet outside 0 0 0 0
aaa authorization include http outside 0 0 0 0
aaa authorization include ftp inside 0 0 0 0
aaa authorization include telnet inside 0 0 0 0
aaa authorization include http inside 0 0 0 0
```

For further information about the different options available for the **authorization** and **authentication** parameters, refer to the *Cisco PIX Firewall Command Reference*.

---

## Enabling Secure Authentication of Web Clients

PIX Firewall Version 6.3 introduces a secured method of exchanging usernames and passwords between a web client and a PIX Firewall by using HTTP over SSL (HTTPS). HTTPS encrypts the user name and password and makes the transmission secure.

Previous versions of PIX Firewall, when authenticating a web browser using a AAA server, obtained the user name and password from the HTTP client in clear text.

Add the following keyword to the **aaa** command to enable this feature:

```
aaa authentication secure-http-client
```

The keyword **secure-http-client** enables this feature so that username and password are exchanged securely between HTTP clients and the PIX Firewall.

To enable this feature, you must configure AAA authentication, using the following command:

```
aaa authentication include authen_service if_name 0 0 0 0 <server_tag|LOCAL>
```

For the syntax of this command see the [“Configuring AAA” section on page 3-8](#).

This feature also supports authentication of clients accessing secure (HTTPS) web sites.



**Note**

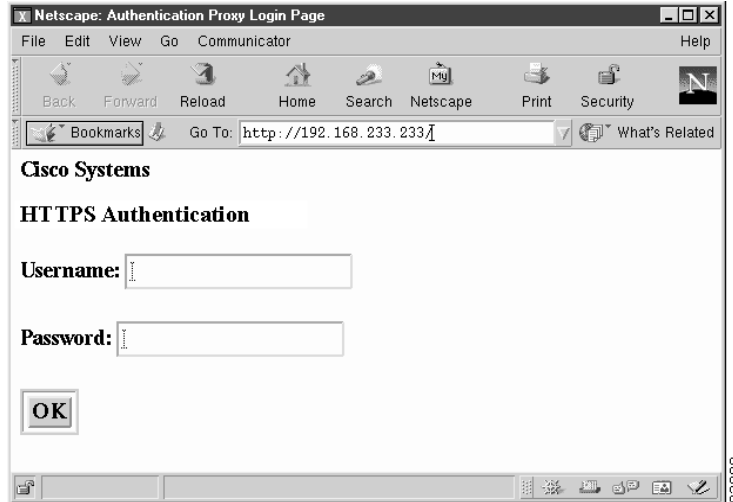
---

Enabling AAA authentication **secure-http-client** is not required to authenticate HTTPS sessions.

---

After enabling this feature, when a user accesses a web page requiring authentication, the PIX Firewall displays the Authentication dialog box shown in [Figure 3-2](#).

Figure 3-2 Secure Authentication Page

**Note**

The Cisco Systems text field shown in this example was customized using the **auth-prompt** command. For the detailed syntax of this command refer to the *Cisco PIX Firewall Command Reference*. If you do not enter a string using the **auth-prompt** command, this field will be blank.

After the user enters a valid username and password, an “Authentication Successful” page appears and closes automatically. If the user fails to enter a valid username and password, an “Authentication Failed” page appears.

A maximum of 16 concurrent HTTPS authentications are allowed. If all 16 HTTPS authentication processes are running, a new connection requiring authentication will not succeed. An authentication process starts when the PIX Firewall receives the user name and password from the browser and ends when it receives the authentication result from the AAA server. The length of time required to complete each authentication process depends on the response time from the authentication source. If the LOCAL database is used, it is very fast, while if a RADIUS or TACACS+ server is used, it will depend on the server response time.

**Note**

Pre-PIX 6.3 configurations that include AAA authentication include `tcp/0..` will inherit the HTTPS Authentication Proxy feature enabled with a code upgrade to PIX 6.3 or later.

When using the **uauth timeout 0** command, HTTPS authentication will not work if a browser initiates multiple TCP connections to get a web page after HTTPS authentication. In this scenario, the first connection is allowed, but the subsequent connections will trigger authentication because the `uauth timeout` is set to 0. As a result, users will be presented authentication pages continuously even though the correct username and password are entered each time. You can avoid this problem by setting the `uauth timeout` to 1 second. However, this opens a 1-second window that could conceivably allow a non-authenticated user to obtain access from the same source IP address.

If a web browser launches an HTTPS web page request while secure authentication is in process for a previous HTTP request, the HTTPS request triggers a second secure authentication process, even if secure authentication is not specifically enabled for HTTPS. Once the authentication process for either web page is completed successful, the remaining request can be completed by reloading the page.

Because HTTPS authentication occurs on the SSL port 443, do not use the **access-list** command to block traffic from the HTTP client to HTTP server on port 443. Also, if you configure static PAT for web traffic on port 80, you must also configure a static entry for SSL port 443.

## Configuring RADIUS Authorization

PIX Firewall allows a RADIUS server to send user group attributes to the PIX Firewall in the RADIUS authentication response message.

The administrator first defines access lists on the PIX Firewall for each user group. For example, there could be access lists for each department in an organization, sales, marketing, engineering, and so on. The administrator then lists the access list in the group profile in the Cisco version of RADIUS, called CiscoSecure.

The PIX Firewall requests authentication of the user by the RADIUS server. If the user is authorized, the RADIUS server returns a confirming authorization response message to the PIX Firewall with vendor specific attribute 11 (filter-id) set to the access list for the given user's group. RADIUS attribute 11 cannot be used to pass this information.

To maintain consistency, PIX Firewall also provides the same functionality for TACACS+.



### Note

---

Access lists can be used with either RADIUS or TACACS but authorizing FTP, HTTP, or Telnet is only possible with TACACS+.

---

To restrict users in a department to three servers and deny everything else, the **access-list** command statements are as follows:

```
access-list eng permit ip any server1 255.255.255.255
access-list eng permit ip any server2 255.255.255.255
access-list eng permit ip any server3 255.255.255.255
access-list eng deny ip any any
```

In this example, the vendor-specific attribute string in the CiscoSecure configuration has been set to **acl=eng**. Use this field in the CiscoSecure configuration to identify the access list identification name. The PIX Firewall gets the **acl=acl\_ID** string from CiscoSecure, extracts the ACL identifier and puts it in the user's uauth entry.

When a user tries to open a connection, PIX Firewall checks the access list in the user's uauth entry, and depending on the permit or deny status of the access list match, permits or denies the connection. When a connection is denied, PIX Firewall generates a corresponding syslog message. If there is no match, then the implicit rule is to deny.

Because the source IP of a given user can vary depending on where they are logging in from, set the source address in the **access-list** command statement to **any**, and the destination address to identify the network services to which user is permitted or denied access.



### Note

---

The **aaa authorization** command does not require a separate RADIUS option.

---

## Using MAC-Based AAA Exemption

PIX Firewall Versions 6.3 and higher let you use Media Access Control (MAC) addresses to bypass authentication for devices, such as Cisco IP Phones, that do not support AAA authentication. To use this feature, you identify the MAC addresses on the inside (higher security) interface. The PIX Firewall bypasses the AAA server for traffic that matches using both the MAC address and the IP address that has been dynamically assigned to the MAC address. Authorization services are automatically disabled when you bypass authentication. Accounting records are still generated (if enabled), but the username is not displayed.

To enable MAC-based AAA exemption, create a list of MAC addresses to be exempted from AAA authentication and then assign the list to a AAA server.

**Note**

---

This feature cannot be applied on the outside or lower security level interface.

---

To define a list of MAC addresses, enter the following command:

```
mac-list mcl-id deny | permit mac mac-mask
```

Enter this command as many times as necessary to define all the MAC addresses you want to add to the list.

Replace *mcl-id* with the identifier of the MAC list. Use the **permit** option to identify the MAC addresses to be exempted from authentication. Use the **deny** option to prevent the bypassing of authentication. Replace *mac* with a partial MAC address that can be used to select a group of devices based on a common portion of the hardware address, such as the vendor ID. Replace *mac-mask* with a mask that identifies the portion of the MAC address that should be used for matching.

For example, the following entry would bypass authentication for a single MAC address:

```
mypix(config)# mac-list adc permit 00a0.c95d.0282 ffff.ffff.ffff
```

In this example, the mask FFFF.FFFF.FFFF instructs the PIX Firewall to match all 12 digits (six bytes) in the preceding hexadecimal address.

The following entry would bypass authentication for all Cisco IP Phones, which have the hardware ID 0003.E3:

```
mypix(config)# mac-list adc permit 0003.E300.0000 FFFF.FF00.0000
```

To apply the MAC list to the AAA server, enter the following command:

```
aaa mac-exempt match mcl-id
```

Replace *mcl-id* with the identifier for the MAC list that you want to apply.

For example, the following command applies the MAC-list *adc* to the AAA server.

```
aaa mac-exempt match adc
```

To view the current entries in a specific MAC list, enter the following command:

```
show mac-list [mcl-id]
```

If you omit the MAC list identifier, the system displays all currently configured MAC lists.

To clear all the entries on a MAC list, enter the following command:

```
clear mac-list [mclid]
```

If you omit the MAC list identifier, the system clears all the currently configured MAC lists.

## Access Control Configuration Example

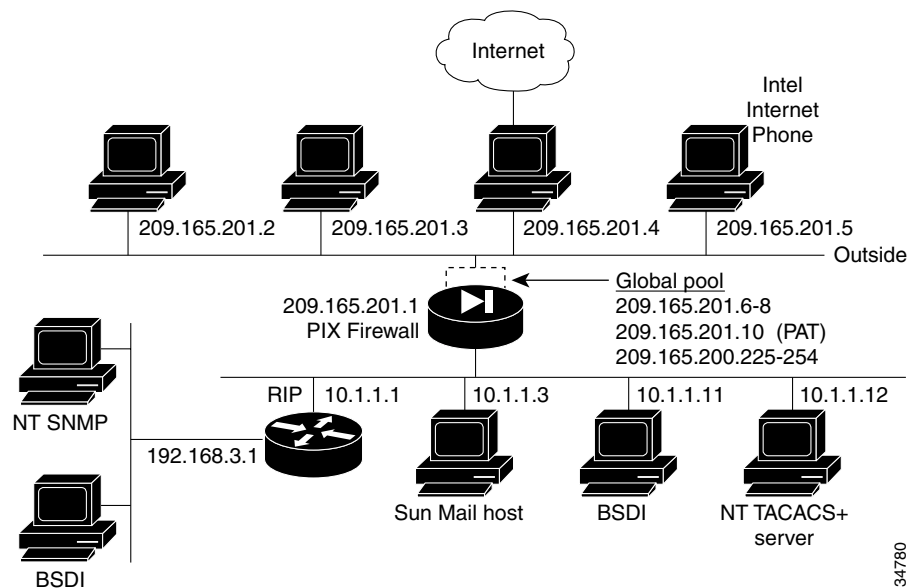
This section provides an example of how to implement access control and includes the following topics:

- [Basic Configuration, page 3-14](#)
- [Authentication and Authorization, page 3-16](#)
- [Managing Access to Services, page 3-16](#)
- [Adding Comments to ACLs, page 3-18](#)

### Basic Configuration

Figure 3-3 illustrates the network configuration used in this example.

**Figure 3-3 Two Interfaces with NAT—Access Control**



34780

The following procedure shows the basic configuration required for this example. This procedure is similar to the configuration shown in “Basic Configuration Examples:” in Chapter 2, “Establishing Connectivity”:

**Step 1** Identify the security level and names of each interface by entering the following commands:

```
nameif ethernet0 outside security0
nameif ethernet1 inside security100
```

**Step 2** Identify the line speed of each interface by entering the following commands:

```
interface ethernet0 100basex
interface ethernet1 100basex
```

You may get better performance by changing the default **auto** option in the **interface** command to the specific line speed for the interface card.

**Step 3** Identify the IP addresses for each interface:

```
ip address inside 10.1.1.1 255.255.255.0
ip address outside 209.165.201.1 255.255.255.224
```

**Step 4** Specify the host name for the PIX Firewall:

```
hostname pixfirewall
```

This name appears in the command-line prompt.

**Step 5** Let inside IP addresses be recognized on the outside network and let inside users start outbound connections:

```
nat (inside) 1 0.0.0.0 0.0.0.0
nat (inside) 2 192.168.3.0 255.255.255.0
global (outside) 1 209.165.201.6-209.165.201.8 netmask 255.255.255.224
global (outside) 1 209.165.201.10 netmask 255.255.255.224
global (outside) 2 209.165.200.225-209.165.200.254 netmask 255.255.255.224
```

**Step 6** Set the outside default route to the router attached to the Internet:

```
route outside 0 0 209.165.201.4 1
```

**Example 3-2** shows the basic configuration required to implement a PIX Firewall with two interfaces with NAT.

### **Example 3-2 Two Interfaces with NAT—Basic Configuration**

```
nameif ethernet0 outside security0
nameif ethernet1 inside security100
interface ethernet0 100basex
interface ethernet1 100basex
ip address inside 10.1.1.1 255.255.255.0
ip address outside 209.165.201.1 255.255.255.224
hostname pixfirewall
nat (inside) 1 0.0.0.0 0.0.0.0
nat (inside) 2 192.168.3.0 255.255.255.0
global (outside) 1 209.165.201.6-209.165.201.8 netmask 255.255.255.224
global (outside) 1 209.165.201.10 netmask 255.255.255.224
global (outside) 2 209.165.200.225-209.165.200.254 netmask 255.255.255.224
route outside 0 0 209.165.201.4 1
```

## Authentication and Authorization

This section describes how to implement authentication and authorization for traffic through the PIX Firewall, using a TACACS+ server. The commands used for this purpose are in addition to the basic firewall configuration required, which is described in the previous section, “[Basic Configuration](#).”

The **aaa-server** command specifies the IP address of the TACACS+ authentication server. The **aaa authentication** command statement specifies that users on network 192.168.3.0 starting FTP, HTTP, and Web connections from the inside interface be prompted for their usernames and passwords before being permitted to access the servers on other interfaces. The **aaa authorization** command statement lets the users on 192.168.3.0 access FTP, HTTP, or Telnet, and any TCP connections to anywhere as authorized by the AAA server. Even though it appears that the **aaa** commands let the PIX Firewall set security policy, the authentication server actually does the work to decide which users are authenticated and what services they can access when authentication is permitted.

[Example 3-3](#) shows the command listing for configuring access to services for the network illustrated in [Figure 3-3](#).

### Example 3-3 Authentication and Authorization Commands

```
aaa-server TACACS+ (inside) host 10.1.1.12 1q2w3e
aaa authentication include ftp inside 192.168.3.0 255.255.255.0 0 0 TACACS+
aaa authorization include ftp inside 192.168.3.0 255.255.255.0 0 0
aaa authentication include http inside 192.168.3.0 255.255.255.0 0 0 TACACS+
aaa authorization include http inside 192.168.3.0 255.255.255.0 0 0
aaa authentication include telnet inside 192.168.3.0 255.255.255.0 0 0 TACACS+
aaa authorization include telnet inside 192.168.3.0 255.255.255.0 0 0
```

## Managing Access to Services



### Note

The commands in this section are used in addition to the basic firewall configuration required, which is described in the previous section, “[Basic Configuration](#).”

The following procedure shows the commands required to manage user access to H.323 and Web services:

**Step 1** Create outbound access lists to determine which hosts can access services:

```
access-list acl_in deny tcp host 192.168.3.3 any eq 1720
access-list acl_in permit tcp host 192.168.3.3 any eq 80
access-list acl_in permit tcp host 10.1.1.11 any eq 80
access-list acl_in deny tcp any any eq 80
```

The first **access-list** command statement denies host 192.168.3.3 from accessing H.323 (port 1720) services such as MS NetMeeting or Intel Internet Phone. The next command statement permits host 192.168.3.3 to use the Web. The third **access-list** command statement permits host 10.1.1.11 access to the Web (at port 80). The last command statement denies all other hosts from accessing the Web (port 80).

- Step 2** Specify that the **access-list** group regulates the activities of inside hosts starting outbound connections:
- ```
access-group acl_in in interface inside
```



Note For information about logging activity associated with specific ACLs, see “[Logging Access Control List Activity](#)” in Chapter 9, “[Accessing and Monitoring PIX Firewall](#).”

- Step 3** Create static address mappings:

```
static (inside, outside) 209.165.201.16 192.168.3.16 netmask 255.255.255.240
```

This example maps IP addresses 209.165.201.17 through 209.165.201.30 to 192.168.3.17 through 192.168.3.30.

- Step 4** Enable VoIP access:

```
access-list acl_out permit tcp any 209.165.201.16 255.255.255.240 eq h323
```

This command lets users on the Internet send Intel Internet Phone requests to users on the protected network. A request can be sent to any IP address in the range from 209.165.201.16 through 209.165.201.31 and the PIX Firewall will translate this address to the next available IP address in the range from 192.168.3.16 through 192.168.3.31.

- Step 5** Establish an externally visible IP address for Web access:

```
static (inside, outside) 209.165.201.11 10.1.1.11
access-list acl_out permit tcp any host 209.165.201.11 eq 80
```

The **static** command statement with the **access-list** command statement establishes an externally visible IP address for Web access (port 80 in the **access-list** command statement).

[Example 3-4](#) shows the command listing for configuring access to services for the network illustrated in [Figure 3-3](#).

Example 3-4 Configuring Access to Services

```
access-list acl_in deny tcp host 192.168.3.3 any eq 1720
access-list acl_in permit tcp host 192.168.3.3 any eq 80
access-list acl_in permit tcp host 10.1.1.11 any eq 80
access-list acl_in deny tcp any any eq 80
access-group acl_in in interface inside
access-list acl_out permit tcp any 209.165.201.16 255.255.255.240 eq h323
static (inside, outside) 209.165.201.11 10.1.1.11
access-list acl_out permit tcp any host 209.165.201.11 eq 80
```

Adding Comments to ACLs

PIX Firewall Version 6.3 and higher lets you include comments about entries in any ACL. The remarks make the ACL easier to understand and scan. A remark can be up to 100 characters and can precede or follow an **access-list** command. However, for clarity, comments should be placed consistently within an access list. There is no run-time performance impact because remarks are stored within an access control entry (ACE) data structure.

Following is the command syntax to specify a comment:

```
access-list acl_id remark text
```

Replace *acl_id* with the ACL identifier and *text* with up to 100 characters of text. If more than 100 characters are entered, it is truncated. The starting position of the text is 1 after the **remark** keyword and leading spaces are allowed. Trailing spaces are ignored.

To remove a remark, precede the command with **no**; trailing spaces in the command line do not affect the matching result.

To allow you to add ACL remarks at the top of an ACL, you can now create an “empty” ACL, containing remarks without any access control entries. When all remarks are removed from this type of ACL, the ACL is also removed.

Using TurboACL

This section describes how to use the TurboACL feature introduced with PIX Firewall Version 6.2. It includes the following topics:

- [Overview, page 3-18](#)
- [Globally Configuring TurboACL, page 3-19](#)
- [Configuring Individual TurboACLs, page 3-19](#)
- [Viewing TurboACL Configuration, page 3-20](#)

Overview

An access list typically consists of multiple access list entries, organized internally by PIX Firewall as a linked list. When a packet is subjected to access list control, the PIX Firewall searches this linked list linearly to find a matching element. The matching element is then examined to determine if the packet is to be transmitted or dropped. With a linear search, the average search time increases proportional to the size of the list.

TurboACL is a feature introduced with PIX Firewall Version 6.2 that improves the average search time for access control lists containing a large number of entries. The TurboACL feature causes the PIX Firewall to compile tables for ACLs and this improves searching of long ACLs.

You can enable this feature for the entire PIX Firewall and then disable it for specific ACLs, or enable it only for specific ACLs. For short ACLs, TurboACL does not improve performance. A TurboACL search, no matter how short the ACL, requires about the same amount of time as a regular ACL search of from twelve to eighteen entries. For this reason, even when enabled, the TurboACL feature is only applied to ACLs with nineteen or more entries.

**Note**

When you add or delete an element from a turbo-enabled ACL the internal data tables associated with the ACL are regenerated, which produces an appreciable load on the PIX Firewall CPU.

The TurboACL feature requires significant amounts of memory and is most appropriate for high-end PIX Firewall models, such as the PIX 525 or PIX 535. The minimum memory required for TurboACL is 2.1 MB and approximately 1 MB of memory is required for every 2000 ACL elements. The actual amount of memory required depends not only on the number of ACL elements but also on the complexity of the entries.

**Note**

With PIX Firewall models having limited memory, such as the PIX 501, implementing the TurboACL feature may cause problems, such as not being able to load Cisco PIX Device Manager (PDM). If memory problems occur after enabling TurboACL, disable it using the **no access-list compiled** command.

Globally Configuring TurboACL

The syntax for enabling TurboACL for the entire PIX Firewall is as follows:

```
access-list compiled
```

This configures TurboACL on all ACLs having 19 or more entries. This command causes the TurboACL process to scan through all existing ACLs. During the scan, it marks and turbo-compiles any ACL which has 19 or more access control entries (ACEs) and has not yet been turbo-compiled.

The command **no access-list compiled**, which is the default, causes the TurboACL process to scan through all compiled ACLs and mark every one as non-turbo. It also deletes all existing TurboACL structures.

When the PIX Firewall is running, the command **access-list compiled** marks every ACL to be turbo-configured, and the command **no access-list compiled** marks every ACL as non-turbo.

Configuring Individual TurboACLs

The individual TurboACL command can be used to enable individual turbo configuration for individual ACLs when TurboACL is not globally enabled. Also, after globally configuring TurboACL, you can disable the turbo-compiled feature for individual ACLs by using the individual TurboACL command. The syntax of this command is as follows.

```
access-list acl_name compiled
```

This command is used to individually enable or disable TurboACL on a specific ACL. The *acl_name* must specify an existing ACL. This command will cause the TurboACL process to mark the ACL specified by *acl_name* to be turbo-compiled if the ACL has 19 or more ACEs and has not yet been turbo-compiled.

If you enter the **no** form of the command, the TurboACL process deletes the TurboACL structures associated with the ACL and marks the ACL as non-turbo.

Viewing TurboACL Configuration

The **show access-list** command displays the memory usage of each individually turbo-compiled ACL and the shared memory usage for all the turbo-compiled ACLs. If no ACL is turbo-compiled, no turbo-statistic is displayed. This command also shows the number of ACEs in an ACL and whether an ACL is configured with TurboACL. Note that an ACL may be configured for turbo but it will not be compiled unless the number of ACEs exceeds the threshold. In such a case, this command will show that the ACL is turbo-configured, but there will not be any entry for the ACL in the TurboACL statistic output.

[Example 3-5](#) provides sample output from the **show access-list** command:

Example 3-5 TurboACL Statistics

```

pix# show access-list
TurboACL statistics:
ACL                State          Memory (KB)
-----
acl_foo            Operational    5
Acl_bar            Operational    2
Shared memory usage: 2046 KB
access-list compiled
access-list acl_foo turbo-configured; 19 elements
access-list acl_foo permit tcp any host 10.0.0.252 (hitcnt=0)
access-list acl_foo permit tcp any host 10.0.0.255 (hitcnt=0)
access-list acl_foo permit tcp any host 10.0.0.253 (hitcnt=0)
access-list acl_foo permit tcp 10.1.0.0 255.0.0.0 host 10.0.0.254 eq telnet (hitcnt=2)
access-list acl_foo permit tcp 10.1.0.0 255.0.0.0 host 10.0.0.254 eq 1 (hitcnt=0)

```

Downloading Access Lists

PIX Firewall supports per-user access list authorization, by which a user is authorized to do only what is permitted in the user's individual access list entries. This section describes how to implement this feature and includes the following topics:

- [Configuring Downloadable ACLs, page 3-20](#)
- [Downloading a Named Access List, page 3-21](#)
- [Downloading an Access List Without a Name, page 3-22](#)
- [Software Restrictions, page 3-23](#)

Configuring Downloadable ACLs

This feature lets you configure per-user access lists on a AAA server and then download the access list to a PIX Firewall during user authentication.

Beginning with PIX Firewall Version 6.2, these access lists can be downloaded from a AAA server and do not need to be configured separately on the PIX Firewall. This feature improves scalability when using access lists for individual users.



Note

Downloadable ACLs are only supported with RADIUS servers and not with TACACS+ servers.

The following are the two methods for downloading an access list from a AAA server to the PIX Firewall:

- Downloading a named access list—Configure a user (real) authentication profile to include a Shared Profile Component (SPC) and then configure the SPC to include the access list name and the actual access list. This method should be used when there are frequent requests for downloading a large access list.
- Downloading an access list without a name—Configure a user authentication profile on a AAA server to include the PIX Firewall access list to be downloaded. This method should be used when there are no frequent requests for the same access list.

Downloading a Named Access List

To download a named access list during a user authentication, the following procedure must be performed on Cisco Secure ACS 3.0 or higher:

-
- Step 1** Select **Downloadable PIX ACLs** from the Shared Profile Component (SPC) menu item.
- Step 2** Click **Add** to add an ACL definition and enter the name, description, and the actual definitions for the ACL.

The ACL definition consists of one or more PIX Firewall **access-list** commands with each command on a separate line. Each command must be entered without the **access-list** keyword and the name for the access list because they are not needed. The rest of the command line must conform to the syntax and semantics rules of the PIX Firewall **access-list** command. A PIX Firewall Syslog message will be logged if there is an error in a downloaded **access-list** command.

The following is an example of an ACL definition before it is downloaded to the PIX Firewall:

```

+-----+
| Shared profile Components
|
|   Downloadable PIX ACLs
|
| Name:                acs_ten_acl
| Description: 10 PIX access-list commands
|
|
|   ACL Definitions
|
| permit tcp any host 10.0.0.254
| permit udp any host 10.0.0.254
| permit icmp any host 10.0.0.254
| permit tcp any host 10.0.0.253
| permit udp any host 10.0.0.253
| permit icmp any host 10.0.0.253
| permit tcp any host 10.0.0.252
| permit udp any host 10.0.0.252
| permit icmp any host 10.0.0.252
| permit ip any any
+-----+

```

- Step 3** Configure a Cisco Secure ACS user or a group through **User Setup** or **Group Setup** to include the defined ACL in the user or group settings.

Once the configuration is properly configured, a user authentication request will first cause the access list name to be sent to the PIX Firewall. The PIX Firewall will determine if the named ACL already exists and if not, the PIX Firewall will request the ACL to be downloaded. A named ACL is not downloaded again as long as it exists on the PIX Firewall.

If the download is successful, the ACL on the PIX Firewall will have the following name:

```
#ACSACL#-acl_name-12345678
```

Where *acl_name* is the name for the access list defined in the SPC and 12345678 is a unique version ID. If the named access list is not configured on ACS or the download fails for any other reason, a Syslog message will be logged.

After the ACL definition has been downloaded to the PIX Firewall, it looks like the following:

```
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit tcp any host 10.0.0.254
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit udp any host 10.0.0.254
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit icmp any host 10.0.0.254
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit tcp any host 10.0.0.253
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit udp any host 10.0.0.253
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit icmp any host 10.0.0.253
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit tcp any host 10.0.0.252
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit udp any host 10.0.0.252
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit icmp any host 10.0.0.252
access-list #ACSACL#-PIX-acs_ten_acl-3b5385f7 permit ip any any
```

- Step 4** Activate the use of downloadable ACLs by performing the following steps:
- Click **Interface Configuration** on the **Cisco Secure ACS** main menu.
 - Click **Advanced Options** on the **Interface Configuration** menu.
 - Select either or both of the following options:
 - User-Level Downloadable ACLs
 - Group-Level Downloadable ACLs

Downloading an Access List Without a Name

To download an access list without using a name during a user authentication, perform the following at a AAA RADIUS server:

Configure CISCO-specific VSA (Attribute 26) string of a user authentication profile in the following format:

```
ip:inacl#nnn=ACL_COMMAND
```

where:

- *ip:inacl#* is the string that specifies an input ACL.
- *nnn* is a number in the range from 0 to 999999999 that identifies the order of the **access-list** command statement to be configured on the PIX Firewall. If this parameter is omitted, the sequence value is 0.
- *ACL_COMMAND* represents one or more PIX Firewall **access-list** commands.

Statements are separated by colons (:). Statements should *not* include the **access-list** command or the access list name. You can configure multiple occurrences of the string “ip:inacl#*nnn*=” in the same user authentication profile to define a PIX Firewall access list. If multiple entries have the same sequence number, they will be configured in the same order as they appear in the Cisco-specific VSA attribute.

Multiple lines may be used to configure multiple elements, but an element must be completely contained on a single line. For example, the **permit tcp any any** command cannot be broken into two separate lines.

A downloadable ACL without a name is assigned a name by the PIX Firewall after it is downloaded in the following format:

```
AAA-user-username
```

Where *username* is the name of the user that is being authenticated.

If an **access-list** command statement has a syntax or semantics error, or if the **no access-list** command is used (an empty access list), Syslog messages will be generated. However, an error with a single **access-list** command does not abort the processing of the entire downloaded ACL.

Example 3-6 Example Configuration for a Downloadable Access List

The following configuration would be entered for the user Admin in the [009\001] cisco-av-pair field under **Group Setup>Cisco IOS/PIX RADIUS Attributes**:

```
ip:inacl#1=permit tcp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
ip:inacl#99=deny tcp any any
ip:inacl#2=permit udp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
ip:inacl#99=deny udp any any
ip:inacl#3=permit icmp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
```

The resulting downloaded **access-list** commands on PIX Firewall are as follows:

```
access-list AAA-user-foo; 5 elements
access-list AAA-user-foo permit tcp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
access-list AAA-user-foo permit udp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
access-list AAA-user-foo permit icmp 10.1.0.0 255.0.0.0 10.0.0.0 255.0.0.0
access-list AAA-user-foo deny tcp any any
access-list AAA-user-foo deny udp any any
```

Software Restrictions

When downloading access lists via RADIUS, the following restrictions apply:

- RADIUS packet size is limited to 4096, but the actual size for the access list can vary considerably depending on the existence of other variable-length, RADIUS attributes.
- Each RADIUS attribute field used to specify access lists is limited to 253 bytes.



Note

If there exists any incompatibility between the PIX Firewall and the Cisco IOS access list, the incompatibility will also exist for the downloaded access list. In other words, an access list defined for PIX Firewall on a AAA server may not be valid if the access list is downloaded to Cisco IOS software, and vice versa.

Simplifying Access Control with Object Grouping

This section describes how to use object grouping, a feature introduced in PIX Firewall Version 6.2, for simplifying complex access control policies. It includes the following topics:

- [How Object Grouping Works, page 3-24](#)
- [Using Subcommand Mode, page 3-25](#)
- [Configuring and Using Object Groups with Access Control, page 3-26](#)
- [Configuring Protocol Object Groups, page 3-28](#)
- [Configuring Network Object Groups, page 3-28](#)
- [Configuring Service Object Groups, page 3-28](#)
- [Configuring ICMP-Type Object Groups, page 3-29](#)
- [Nesting Object Groups, page 3-29](#)
- [Displaying Configured Object Groups, page 3-30](#)
- [Removing Object Groups, page 3-30](#)

How Object Grouping Works

Object grouping provides a way to reduce the number of access rules required to describe complex security policies. An access rule can apply to the following types of objects:

- Client host—Makes HTTP, Telnet, FTP, Voice over IP, and other service requests
- Server host—Responds to service requests
- Service type—Services are assigned to well-known, dynamically assigned, or secondary channel TCP or UDP ports
- Subnet—The network address of internal or external subnetworks where server or client hosts are located
- ICMP types—Such as ECHO-REPLY

An access rule allows or denies traffic matching a specific combination of these objects. For example, an access rule might cause the PIX Firewall to allow a designated client to access a particular server host for a specific service. When there is only one client, one host, and one service, only one access rule is needed. However, as the number of clients, servers, and services increases, the number of rules required may increase exponentially.

Object grouping provides a way to group objects of a similar type into a group so that a single access rule can apply to all the objects in the group. For example, consider the following three object groups:

- MyServices—Includes the TCP/UDP port numbers of the service requests that are allowed access to the internal network
- TrustedHosts—Includes the host and network addresses allowed access to the greatest range of services and servers
- PublicServers—Includes the host addresses of servers to which the greatest access is provided

After creating these groups, you could use a single access rule to allow trusted hosts to make specific service requests to a group of public servers. Object groups can also contain other object groups or be contained by other object groups.

Object grouping dramatically compresses the number of access rules required to implement a particular security policy. For example, a customer policy that required 3300 access rules might only require 40 rules after hosts and services are properly grouped.

Using Subcommand Mode

The general syntax of the **object-group** command is as follows:

```
object-group object-type grp-id
```

Replace *object-type* with one of the following object types:

- **protocol**—Group of IP protocols. It can be one of the keywords **icmp**, **ip**, **tcp**, or **udp**, or an integer in the range 1 to 254 representing an IP protocol number. To match any Internet protocol, including ICMP, TCP, and UDP, use the keyword **ip**.
- **service**—Group of TCP or UDP port numbers assigned to different services.
- **icmp-type**—Group of ICMP message types to which you permit or deny access.
- **network**—Group of hosts or subnets

Replace *grp-id* with a descriptive name for the group.

When you enter the **object-group** command, the prompt changes to the subcommand mode appropriate for the type of object. Commands entered in the subcommand mode apply to the object type and group name identified in the **object-group** command.

The prompts in each subcommand mode are as follows:

```
pix(config-protocol)#  
pix(config-service)#  
pix(config-icmp-type)#  
pix(config-network)#
```

Enter a question mark (?) in the subcommand mode to view the permitted subcommands.

In subcommand mode, you can enter object grouping subcommands as well as all other PIX Firewall commands including **show** commands and **clear** commands. When you enter any valid configuration command, such as **access-list**, the subcommand mode is terminated. You can also terminate the subcommand mode by entering the **exit** or **quit** commands. Subcommands are indented when they are shown or saved by any of the following commands:

- **show config**
- **write**
- **config**

Configuring and Using Object Groups with Access Control

To configure an object group and to use it for configuring access lists, perform the following steps:

Step 1 Enter the appropriate subcommand mode for the type of group you want to configure.

The syntax of the **object-group** command is as follows:

```
pix(config)# object-group {protocol|network|icmp-type} grp-id
pix(config)# object-group service grp-id {tcp|udp|tcp-udp}
```

Use the first parameter to identify the type of object group you want to configure. Replace the second parameter *grp-id* with a descriptive name for the group. When you enter the **object-group** command, the system enters the appropriate subcommand mode for the type of object you are configuring.

For example, the following command identifies an object group containing trusted hosts:

```
pix(config)# object-group network TrustedHosts
```

When you enter this command, the system enters the network object subcommand mode and the PIX Firewall system prompt appears as follows:

```
pix(config-network)#
```

All subcommands entered from this prompt apply to the object group identified by the **object-group** command. In this example, the object group name is `TrustedHosts`.

Step 2 Define the members of the object group.

Use the subcommands permitted within the subcommand mode to define members of the object group. Use the **group-object** subcommand to add a subgroup within the current object group.

For example:

```
pix(config)# object-group network ftp_servers
pix(config-network)# network-object host 209.165.201.3
pix(config-network)# network-object host 209.165.201.4
pix(config-network)# exit
pix(config)# object-group network TrustedHosts
pix(config-network)# network-object host sjc.eng.ftp
pix(config-network)# network-object host 209.165.201.1
pix(config-network)# network-object 192.168.1.0 255.255.255.0
pix(config-network)# group-object ftp_servers
```

These commands add the following objects to the group `TrustedHosts`:

- One host by host name
- One host by network address
- One subnetwork
- One subgroup (**ftp_servers**)

Step 3 (Optional) Describe the object group by entering the following command from the subcommand mode:

```
pix(config-network)# description text
```

This command lets you add a description of up to 200 characters to an object group. Replace *text* with the descriptive information you wish to enter.

Step 4 Return to configuration mode by entering the following command:

```
pix(config-network)# exit
```

Step 5 (Optional) Verify that the object group has been configured successfully:

```
pix(config)# show object-group [network | services | icmp-type] [grp-id]
```

This command displays a list of the currently configured object groups of the specified type. Without a parameter, the command displays all object groups.

For example:

```
pix(config)# show object-group
object-group network ftp_servers
  description: This is a group of FTP servers
  network-object host 209.165.201.3
  network-object host 209.165.201.4
object-group network TrustedHosts
  network-object host 209.165.201.1
  network-object 192.168.1.0 255.255.255.0
  group-object ftp_servers
```

Step 6 Apply the **access-list** command to the object group.



Note Beginning with Version 5.3, the PIX Firewall uses access lists to control connections between inside and outside networks. Access lists are implemented with the **access-list** and **access-group** commands. These commands are used instead of the **conduit** and **outbound** commands, which were used in earlier versions of PIX Firewall. In PIX Firewall software releases later than Version 6.3, the **conduit** and **outbound** commands are no longer supported. To help you with the conversion process, a tool is available online at: <https://cco-dev.cisco.com/cgi-bin/Support/OutputInterpreter/home.pl>.

Replace the parameters of the **access-list** commands with the corresponding object group:

- Replace the **protocol** parameter with a protocol object group.
- Replace local and remote IP addresses and subnet masks with a network object group.
- Replace the **port** parameter with a service object group.
- Replace the **icmp-type** parameter with an icmp-type object group.



Note Empty object groups cannot be used with any commands.

For example, the following command permits access to the members of the object group **TrustedHosts**:

```
pix(config)# access-list acl permit tcp object-group TrustedHosts host 1.1.1.1
```

Refer to the **access-list** commands in the *Cisco PIX Firewall Command Reference* for the detailed syntax of these commands.

Step 7 (Optional) Use the **show access-list** command to display the expanded access list entries:

```
pix(config)# show access-list
access-list acl permit tcp host 209.165.201.1 host 1.1.1.1
access-list acl permit tcp 192.168.1.0 255.255.255.0 host 1.1.1.1
access-list acl permit tcp host 209.165.201.3 host 1.1.1.1
access-list acl permit tcp host 209.165.201.4 host 1.1.1.1
```



Note The **show config** and **write** commands display the commands in the same way they are configured.

Configuring Protocol Object Groups

This section describes the commands required to configure a protocol object group.

Enter the following command to enable the protocol object subcommand mode:

```
pix(config)# object-group protocol grp-id
```

Enter the following command to add a single protocol to the current protocol object group:

```
pix(config-protocol)# protocol-object protocol
```

Replace *protocol* with the numeric identifier of the specific IP protocol (1 to 254) or a literal keyword identifier (**icmp**, **tcp**, or **udp**). If you wish to include all IP protocols, use the keyword **ip**.

Enter the following command to add the object group identified by *grp-id* to the current protocol object group:

```
pix(config-protocol)# group-object grp-id
```

Configuring Network Object Groups

This section describes the commands required to configure a network object group.

Enter the following command to enable the network object subcommand mode:

```
pix(config)# object-group network grp-id
```

Enter the following command to add a single host name or IP address (with subnetwork mask) to the current network object group:

```
pix(config-network)# network-object host host_addr | net_addr netmask
```

Replace *host_addr* with the IP address of the host you are adding to the group. Replace *net_addr* and *netmask* with the network number and subnet mask for a subnetwork.

Enter the following command to add the object group identified by *grp-id* to the current protocol object group:

```
pix(config-network)# group-object grp-id
```

Configuring Service Object Groups

This section describes the commands required to configure a service object group.

Enter the following command to enable the service object subcommand mode:

```
pix(config)# object-group service {tcp|udp|tcp-udp}
```

Enter the following command to add a single TCP or UDP port number to the service object group:

```
pix(config-service)# port-object eq service grp-id
```

Enter the following command to add a range of TCP or UDP port numbers to the service object group:

```
pix(config-service)# port-object range begin_service end_service
```

Enter the following command to add the object group identified by *grp-id* to the current service object group:

```
pix(config-service)# group-object grp-id
```

Configuring ICMP-Type Object Groups

This section describes the commands required to configure an icmp-type object group.

Enter the following command to enable the icmp-type object subcommand mode:

```
pix(config)# object-group icmp-type grp-id
```

Enter the following command to add an ICMP type to the service object group:

```
pix(config-icmp-type)# icmp-object icmp-type
```

Replace *icmp-type* with a numeric value. Refer to the **access-list** command in the *Cisco PIX Firewall Command Reference* for a definition of the permitted values.

Enter the following command to add the object group identified by *grp-id* to the current icmp-type object group:

```
pix(config-icmp-type)# group-object grp-id
```

Nesting Object Groups

The **object-group** command allows logical grouping of the same type of objects and construction of hierarchical object groups for structured configuration. To nest an object group within another object group, perform the following steps:

-
- Step 1** Assign a group ID to the object group that you want to nest within another object group, as in the following example:

```
pix(config)# object-group protocol Group_A
```

- Step 2** Add the appropriate type of objects to the object group:

```
pix(config-protocol)# protocol-object 1
pix(config-protocol)# protocol-object 2
pix(config-protocol)# protocol-object 3
```

- Step 3** Assign a group identifier to the object group within which you want to nest another object group:

```
pix(config)# object-group protocol Group_B
```

Step 4 Add the first object group to the group that will contain that object:

```
pix(config-protocol)# group-object A
```

Step 5 Add any other objects to the group that are required:

```
pix(config-protocol)# protocol-object 4
```

The resulting configuration of Group_B in this example is equivalent to the following:

```
pix(config-protocol)# protocol-object 1
pix(config-protocol)# protocol-object 2
pix(config-protocol)# protocol-object 3
pix(config-protocol)# protocol-object 4
```

Displaying Configured Object Groups

To display a list of the currently configured object groups, use the **show object-group** command:

```
show object-group [ protocol | network | service | icmp-type ] [ id grp_id]
```

Use the listed parameters to restrict the display to specific object types or to identify a specific object group by name. The system displays a list of the currently configured object groups identified by the command. Replace *grp_id* with the name of a specific object group. If you enter the command without any parameters, the system displays all configured object groups.

[Example 3-7](#) shows sample output from the **show object-group** command.

Example 3-7 Show object-group Command Output

```
pix(config)# show object-group
object-group network ftp_servers
  description: This is a group of FTP servers
  network-object host 209.165.201.3
  network-object host 209.165.201.4
object-group network TrustedHosts
  network-object host 209.165.201.1
  network-object 192.168.1.0 255.255.255.0
group-object ftp_servers
```

Removing Object Groups

To remove the object group configuration for all the groups of a specific type, use the **clear object-group** command:

```
pix(config)# clear object-group [protocol | network | services | icmp-type]
```

If you enter the **clear object-group** command without any parameters, the system removes all configured object groups.

To remove a specific object group, use the following command:

```
pix(config)# no object-group grp_id
```

Replace *grp_id* with the identifier assigned to the specific group you want to remove.

**Note**

You cannot remove an object group or make an object group empty if it is used in a command.

Filtering Outbound Connections

This section describes ways to filter web traffic to reduce security risks or inappropriate use and includes the following topics:

- [Filtering ActiveX Objects, page 3-31](#)
- [Filtering Java Applets, page 3-32](#)
- [Filtering URLs with Internet Filtering Servers, page 3-32](#)

ActiveX objects and Java applets may pose security risks because they can contain code intended to attack hosts and servers on a protected network. You can disable ActiveX objects and remove Java applets with the PIX Firewall **filter** command.

You can use the **filter** command to work with a URL filtering server to remove URLs that are inappropriate for use at your site.

Filtering ActiveX Objects

ActiveX controls, formerly known as OLE or OCX controls, are components you can insert in a web page or other application. These controls include custom forms, calendars, or any of the extensive third-party forms for gathering or displaying information. As a technology, ActiveX creates many potential problems for network clients including causing workstations to fail, introducing network security problems, or being used to attack servers.

The syntax of the command for filtering ActiveX objects is as follows:

```
filteractivex port[-port] |except local_ip mask foreign_ip mask
```

This command blocks the HTML <object> commands by commenting them out within the HTML web page. This functionality has been added to the **filter** command with the **activex** option.

**Note**

The <object> tag is also used for Java applets, image files, and multimedia objects, which will also be blocked by the new command.

If the <object> or </object> HTML tags split across network packets or if the code in the tags is longer than the number of bytes in the MTU, PIX Firewall cannot block the tag.

Java and ActiveX filtering of HTML files are performed by selectively replacing the <APPLET> and </APPLET> and <OBJECT CLASSID> and </OBJECT> tags with comments. Filtering of nested tags is supported by converting top-level tags to comments.

**Note**

ActiveX blocking does not occur when users access an IP address referenced by the **alias** command.

Filtering Java Applets

The **filter java** command filters out Java applets that return to the PIX Firewall from an outbound connection. The user still receives the HTML page, but the web page source for the applet is commented out so that the applet cannot execute. The syntax of the command for filtering Java applets is as follows:

```
filter java port[-port] local_ip mask foreign_ip mask
```

Use 0 for the *local_ip* or *foreign_ip* IP addresses to mean all hosts.



Note

If Java applets are known to be in <object> tags, use the **filteractivex** command to remove them.

Examples

To specify that all outbound connections have Java applet blocking, use the following command:

```
filter java 80 0 0 0 0
```

This command specifies that the Java applet blocking applies to web traffic on port 80 from any local host and for connections to any foreign host. To block downloading of Java applets to a host on a protected network, enter a command like the following:

```
filter java http 192.168.3.3 255.255.255.255 0 0
```

This command prevents host 192.168.3.3 from downloading Java applets.

Filtering URLs with Internet Filtering Servers

This section describes how to enable URL filtering. It contains the following topics:

- [Overview, page 3-32](#)
- [Identifying the Filtering Server, page 3-33](#)
- [Buffering HTTP Replies for Filtered URLs, page 3-34](#)
- [Filtering Long URLs with the Websense Filtering Server, page 3-34](#)
- [Filtering HTTPS and FTP Sites, page 3-34](#)
- [Configuring Filtering Policy, page 3-35](#)
- [Filtering Long URLs, page 3-36](#)
- [Viewing Filtering Statistics and Configuration, page 3-36](#)
- [Configuration Procedure, page 3-38](#)

Overview

The **filter url** command lets you designate webs traffic that is to be filtered using one of the following URL filtering applications:

- Websense Enterprise web filtering application—Supported by PIX Firewall Version 5.3 or higher
- Filtering by N2H2 for IFP-enabled devices—Supported by PIX Firewall Version 6.2 or higher

When a user issues an HTTP request to a website, the PIX Firewall sends the request to the web server and to the filtering server at the same time. If the filtering server permits the connection, the PIX Firewall allows the reply from the website to reach the user who issued the original request. If the filtering server denies the connection, the PIX Firewall redirects the user to a block page, indicating that access was denied. The PIX Firewall sends an authenticated user name, a source IP address, and a destination IP address to the filtering server for URL validation and logging purposes.

**Note**

URL filtering only may considerably increase access times to web sites when the filtering server is remote from the PIX Firewall.

Identifying the Filtering Server

You identify the address of the filtering server using the form of the **url-server** command appropriate for the type of filtering server you are using.

For Websense:

```
pix(config)# url-server [(if_name)] host local_ip [timeout seconds] [protocol TCP [version 1 | 4] | UDP]
```

For N2H2:

```
pix(config)# url-server [(if_name)] vendor n2h2 host local_ip[:port number] [timeout <seconds>] [protocol TCP | UDP]
```

Replace *if_name* with the name of the PIX Firewall interface on which you are enabling filtering. Enclose the interface name within parentheses, as in the following example:

```
url-server (inside) host 192.168.1.1
```

By default, if you do not include this parameter, filtering will apply to the inside interface.

Replace *local_ip* with the IP address of the filtering server. Replace *seconds* with the number of seconds the PIX Firewall should wait before giving up on connecting to the filtering server.

Use the protocol option to identify whether you want to use TCP or UDP. With a Websense server, you can also specify the version of TCP you want to use. TCP version 1 is the default. TCP version 4 allows the PIX Firewall to send authenticated usernames and URL logging information to the Websense server, if the PIX Firewall has already authenticated the user.

**Note**

URL filtering may considerably increase access times to web sites when the filtering server is remote from the PIX Firewall.

You can identify more than one filtering server by entering the **url-server** command multiple times. The primary filtering server is the first server that you identify. If you want to change your primary server, use the **no url-server** command with the address of your primary filtering server. Then issue the **url-server** command with the address of your primary server.

**Note**

If you switch the url-server type after configuration, the existing url-server configurations are dropped and you must reenter the configuration for the new filtering server type.

Buffering HTTP Replies for Filtered URLs

By default, when a user issues a request to connect to a specific website, the PIX Firewall sends the request to the web server and to the filtering server at the same time. If the filtering server does not respond before the web content server, the response from the web server is dropped. This delays the web server response from the point of view of the web client.

By enabling the HTTP response buffer, replies from web content servers are buffered and the responses will be forwarded to the requesting user if the filtering server allows the connection. This prevents the delay that may otherwise occur.

To enable the HTTP response buffer, enter the following command:

```
url-block block block-buffer-limit
```

Replace *block-buffer-limit* with the maximum number of blocks that will be buffered. The permitted values are from 0 to 128, which specifies the number of 1550-byte blocks that can be buffered at one time.

Filtering Long URLs with the Websense Filtering Server



Note

PIX Firewall Versions 6.2 and higher support a fixed, maximum URL length of 1159 bytes for the N2H2 filtering server.

To increase the maximum length of a single URL that can be sent to a Websense filtering server, enter the following command:

```
url-block url-size long-url-size
```

Replace *long-url-size* with a value from 2 to 4 for a maximum URL size of 2 KB to 4 KB.

To configure the maximum memory available for buffering long URLs, enter the following command:

```
url-block url-mempool memory-pool-size
```

Replace *memory-pool-size* with a value from 2 to 10240 for a maximum memory allocation of 2 KB to 10 MB.

Filtering HTTPS and FTP Sites

PIX Firewall Version 6.3 introduces support for filtering of HTTPS and FTP sites for Websense filtering servers.



Note

HTTPS and FTP filtering are not supported for the N2H2 filtering server.

HTTPS filtering works by preventing the completion of SSL connection negotiation if the site is not allowed. The browser displays an error message such as “The Page or the content cannot be displayed.”

Because HTTPS content is encrypted, PIX Firewall sends the URL lookup without directory and filename information.

To enable HTTPS filtering, use the following command:

```
filter https dest_port |except localIP local_mask foreign_IP foreign_mask [allow]
```

To enable FTP filtering, use the following command:

```
filter ftp dest_port |except localIP local_mask foreign_IP foreign_mask [allow]  
[interact-block]
```

The **filter ftp** command lets you identify the FTP traffic to be filtered by a Websense server. FTP filtering is not supported on N2H2 servers.

After enabling this feature, when a user issues an FTP GET request to a server, the PIX Firewall sends the request to the FTP server and to the Websense server at the same time. If the Websense server permits the connection, the firewall allows the successful FTP return code to reach the user unchanged. For example, a successful return code is “250: CWD command successful.”

If the Websense server denies the connection, the PIX Firewall alters the FTP return code to show that the connection was denied. For example, the PIX Firewall would change code 250 to “code 550: Directory not found.” Websense only filters FTP GET commands and not PUT commands).

Use the **interactive-block** option to prevent interactive FTP sessions that do not provide the entire directory path. An interactive FTP client allows the user to change directories without typing the entire path. For example, the user might enter **cd ./files** instead of **cd /public/files**.

You must identify and enable the URL filtering server before using these commands. If all URL filtering servers are removed, any associated filtering commands are also removed.

Configuring Filtering Policy

Use the **filter url** command to configure the policy for filtering URLs. The syntax of the command for filtering URLs is as follows.

```
filter url port[-port] local_ip local_mask foreign_ip foreign_mask] [allow] [proxy-block]
```

Replace *port* with the port number on which to filter HTTP traffic. To identify a range of port numbers, enter the beginning and end of the range separated by a hyphen.

To identify specific HTTP traffic for filtering, replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork making requests. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork responding to requests.

With filtering enabled, the PIX Firewall stops outbound HTTP traffic until a filtering server permits the connection. If the primary filtering server does not respond, the PIX Firewall directs the filtering request to the secondary filtering server. The **allow** option causes the PIX Firewall to forward HTTP traffic without filtering when the primary filtering server is unavailable.

Use the **proxy-block** command to drop all requests to proxy servers.

If you want to make exceptions to the general filtering policy, use the following command:

```
filter url except local_ip local_mask foreign_ip foreign_mask]
```

Replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork that you want to exempt from filtering restrictions. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork that you want to exempt from filtering restrictions.

Filtering Long URLs

PIX Firewall Version 6.1 and earlier versions do not support filtering URLs longer than 1159 bytes. PIX Firewall Versions 6.2 and higher support filtering URLs up to 4 KB for the Websense filtering server. PIX Firewall Versions 6.2 and higher support a maximum URL length of 1159 bytes for the N2H2 filtering server.

In addition, PIX Firewall Version 6.2 introduces the **longurl-truncate** and **cgi-truncate** commands to allow handling of URL requests longer than the maximum permitted size. The format for these options is as follows:

```
filter url [http | port[-port] local_ip local_mask foreign_ip foreign_mask] [allow]
[proxy-block] [longurl-truncate | longurl-deny] [cgi-truncate]
```

PIX Firewall Versions 6.2 and higher support a maximum URL length of 1159 bytes for the N2H2 filtering server. Filtering of URLs up to 4 KB is supported for the Websense filtering server. If a URL is longer than the maximum, and you do not enable the **longurl-truncate** or **longurl-deny** options, the firewall drops the packet.

The **longurl-truncate** option causes the PIX Firewall to send only the host name or IP address portion of the URL for evaluation to the filtering server when the URL is longer than the maximum length permitted. Use the **longurl-deny** option to deny outbound URL traffic if the URL is longer than the maximum permitted.

Use the **cgi-truncate** option to truncate CGI URLs to include only the CGI script location and the script name without any parameters. Many long HTTP requests are CGI requests. If the parameters list is very long, waiting and sending the complete CGI request including the parameter list can use up memory resources and affect firewall performance.

Viewing Filtering Statistics and Configuration

Use the commands in this section to view URL filtering information:

To show information about the filtering server, enter the following command:

```
show url-server
```

The following is sample output from this command:

```
url-server (outside) vendor n2h2 host 128.107.254.202 port 4005 timeout 5 protocol TCP
```

To show the URL statistics, enter the following command:

```
show url-server stats
```

The following is sample output from this command:

```
URL Server Statistics:
-----
Vendor                                websense
URLs total/allowed/denied             0/0/0
HTTPSs total/allowed/denied           0/0/0
FTPs total/allowed/denied              0/0/0
```

```

URL Server Status:
-----
10.130.28.18          UP

URL Packets Sent and Received Stats:
-----
Message                Sent      Received
STATUS_REQUEST         65155    34773
LOOKUP_REQUEST         0         0
LOG_REQUEST            0         NA
-----

```

To show URL caching statistics, enter the following command:

```
show url-cache stat
```

The following is sample output from this command:

```

pix(config)# show url-cache stats
URL Filter Cache Stats
-----
      Size :      128KB
  Entries :      1724
    In Use :         0
  Lookups  :         0
     Hits  :         0

```

To show URL filtering performance statistics, enter the following command:

```
show perfmon
```

The following is sample output from this command:

```

pix(config)# show perfmon

PERFMON STATS:      Current      Average
Xlates              0/s          0/s
Connections         0/s          2/s
TCP Conns           0/s          2/s
UDP Conns           0/s          0/s
URL Access          0/s          2/s
URL Server Req     0/s          3/s
TCP Fixup           0/s          0/s
TCPIntercept       0/s          0/s
HTTP Fixup          0/s          3/s
FTP Fixup           0/s          0/s
AAA Authen         0/s          0/s
AAA Author          0/s          0/s
AAA Account         0/s          0/s

```

To show filtering configuration, enter the following command:

```
show filter
```

The following is sample output from this command:

```

pix(config)# show filter
filter url http 0.0.0.0 0.0.0.0 0.0.0.0 0.0.0.0

```

Configuration Procedure

Perform the following steps to filter URLs:

Step 1 Identify the address of the filtering server with the **url-server** commands:

For Websense:

```
# url-server [(if_name)] host local_ip [timeout seconds] [protocol TCP | UDP version 1|4]
```

For N2H2:

```
# url-server [(if_name)] vendor n2h2 host local_ip[:port number] [timeout seconds]
[protocol TCP | UDP]
```

Replace *if_name* with the name of the PIX Firewall interface that is connected to the filtering server (the default is **inside**). Replace *local_ip* with the IP address of the filtering server. Replace *seconds* with the number of seconds the PIX Firewall should wait before giving up on connecting to the filtering server.



Note The default port is 4005. This is the default port used by the N2H2 server to communicate to the PIX Firewall via TCP or UDP. For information on changing the default port, please refer to the *Filtering by N2H2 Administrator's Guide*.

For example:

```
url-server (perimeter) host 10.0.1.1
url-server (perimeter) vendor n2h2 host 10.0.1.1
```

The first command identifies a Websense filtering server with the IP address 10.0.1.1 on a perimeter interface of the PIX Firewall. The second command identifies an N2H2 server at the same interface and address.

Step 2 Configure your filtering policy with the following command:

```
filter url [http | port[-port] local_ip local_mask foreign_ip foreign_mask] [allow]
[proxy-block]
```

Replace *port* with one or more port numbers if a different port than the default port for HTTP (80) is used. Replace *local_ip* and *local_mask* with the IP address and subnet mask of a user or subnetwork making requests. Replace *foreign_ip* and *foreign_mask* with the IP address and subnet mask of a server or subnetwork responding to requests.

The **allow** option causes the PIX Firewall to forward HTTP traffic without filtering when the primary filtering server is unavailable. Use the **proxy-block** command to drop all requests to proxy servers.

For example:

```
filter url http 0 0 0 0
filter url except 10.0.2.54 255.255.255.255 0 0
```

The first command filters all HTTP traffic. The second command exempts all requests from 10.0.2.54 from filtering restrictions.



Note Step 3 through Step 6 only work with PIX Firewall Version 6.2 or higher. Buffering URLs longer than 1159 bytes is only supported for the Websense filtering server.

- Step 3** (Optional) Enable buffering of HTTP replies for URLs that are pending a response from the filtering server by entering the following command:

```
url-block block block-buffer-limit
```

Replace *block-buffer-limit* with the maximum number of blocks that will be buffered.

- Step 4** (Optional) Configure the maximum memory available for buffering pending URLs (and for buffering long URLs with Websense) with the following command:

```
url-block url-mempool memory-pool-size
```

Replace *memory-pool-size* with a value from 2 to 10240 for a maximum memory allocation of 2 KB to 10 MB.

- Step 5** (Optional for Websense only) Configure the maximum size of a single URL with the following command:

```
url-block url-size long-url-size
```

Replace *long-url-size* with a value from 2 to 4 for a maximum URL size of 2 KB to 4 KB. The default value is 2.

- Step 6** (Optional) To handle URLs that are longer than the maximum available buffer size, enter the **filter** command in the following form:

```
filter url [longurl-truncate | longurl-deny | cgi-truncate]
```

Use the **longurl-truncate** command to send only the host name or IP address portion of the URL for evaluation to the filtering server when the URL is longer than the maximum length permitted.

Use the **longurl-deny** option to deny outbound traffic if the URL is longer than the maximum permitted (1159 for N2H2 or configurable up to 4 KB for Websense).

Use the **cgi-truncate** option to send a CGI script as the URL.

- Step 7** (Optional) To display memory usage, enter the following commands:

```
show chunk
show memory
```

- Step 8** (Optional) Use the **url-cache** command if needed to improve throughput, as follows:

```
url-cache dst | src_dst size
```



Note This command does not update Websense logs, which may affect Websense accounting reports. Accumulate Websense run logs before using the **url-cache** command.

Replace *size* with a value for the cache size within the range 1 to 128 (KB).

Use the **dst** keyword to cache entries based on the URL destination address. Select this mode if all users share the same URL filtering policy on the Websense server.

Use the **src_dst** keyword to cache entries based on both the source address initiating the URL request as well as the URL destination address. Select this mode if users do not share the same URL filtering policy on the Websense server.

Step 9 Configure the required URL filters at the user interface of the filtering server.

For more information about filtering with the N2H2 or Websense filtering servers, refer to the following web sites:

<http://www.websense.com>

<http://www.n2h2.com>

Step 10 Use the following commands to view URL filtering information:

To show URL caching statistics, enter the following command:

```
show url-cache stats
```

To show URL filtering performance statistics, enter the following command:

```
show perfmon
```

To show the information about the filtering server, enter the following command:

```
show url-server
```

To show filtering configuration, enter the following command:

```
show filter
```
