6AOSCG0065-29B
March 2016

ADTRAN

Configuration Guide

IPv6 Firewall Protection in AOS

This configuration guide is designed to provide you with an understanding of the Internet Protocol version 6 (IPv6) firewall protection provided by your ADTRAN Operating System (AOS) product. Included in this guide is an overview describing the types of IPv6 firewall protection, detailed command descriptions, and several network examples to cover a variety of firewall applications. The Troubleshooting section provides methods to identify and verify problems with your configuration.

This guide provides IPv6 firewall configuration and troubleshooting instructions using the command line interface (CLI). This guide consists of the following sections:

Overview

A firewall is a collection of components configured to enforce specific access control parameters between your internal (trusted) network and any other (untrusted) network, such as the Internet. A firewall filters inbound and outbound packets to ensure that only authorized packets pass. The firewall can filter packets at several Open Systems Interconnection (OSI) levels and can enforce complex, customized policies.

This guide explains the firewall components in detail, as well as how to configure your AOS product for firewall protection in several different network scenarios. The following sections provide additional information about the components used in enforcing your firewall protection:

- Stateful inspection firewall (refer to *Stateful Inspection Firewall on page 2*)
- Stateless processing (refer to *Stateless Processing on page 6*)
- Allowing specific application traffic (refer to *Application-Specific Processing on page 7*)
- Protection from known network attacks (refer to *Attack Checking on page 7*)
- Static filters (refer to *Static Filters on page 10*)
- Network traffic management when used in conjunction with access control lists (ACLs) and access control policies (ACPs) (refer to *IPv6 Access Control Lists and Access Control Policies on page 11*)

Stateful Inspection Firewall

Contrary to using access groups with the firewall disabled, the IPv6 firewall, when enabled, performs stateful inspection and attack checks on traffic to and through the unit. Every interface has a single access control policy (ACP), whether defined explicitly or the default ACP, which inspects traffic entering that interface. Typically, an IPv6 ACP is configured on the internal interface that permits hosts to initiate traffic to the Internet. A second IPv6 ACP is configured on the public interface that discards traffic initiated from the Internet. The AOS stateful inspection firewall creates an association for a session initiated from the internal network and stores it in an internal database. When the server on the Internet sends a response back to the host, the AOS stateful inspection firewall recognizes that this traffic is associated with an allowed session and allows the traffic to pass through. Since the firewall has detailed knowledge about the current state of every session flowing through the device, it is much more difficult for an attacker to generate traffic that is not blocked by the firewall. Refer to *IPv6 Access Control Policies on page 13* for more information on IPv6 ACPs.

Firewall Associations and Policy Sessions

Firewall associations are needed so that the AOS device does not have to perform the matching process for every packet that comes through the IPv6 firewall. The benefit of firewall associations is that they increase the speed of packet throughput. For example, if the AOS device is processing a long stream of packets that share the same protocol, source, and destination, and the same action should be performed on each packet, then rigorously matching every packet would be needlessly redundant and waste time. When the IPv6 firewall receives the first packet in a unique stream, it calculates the appropriate action to perform on the packet. Through the use of firewall associations, the AOS device can simply look at subsequent packets in a stream, note that the packets look the same as the first, and perform the calculated action upon them.

Active Policy Sessions

Active policy sessions are firewall associations that are currently alive, thus they allow faster processing of appropriate packets as described previously.

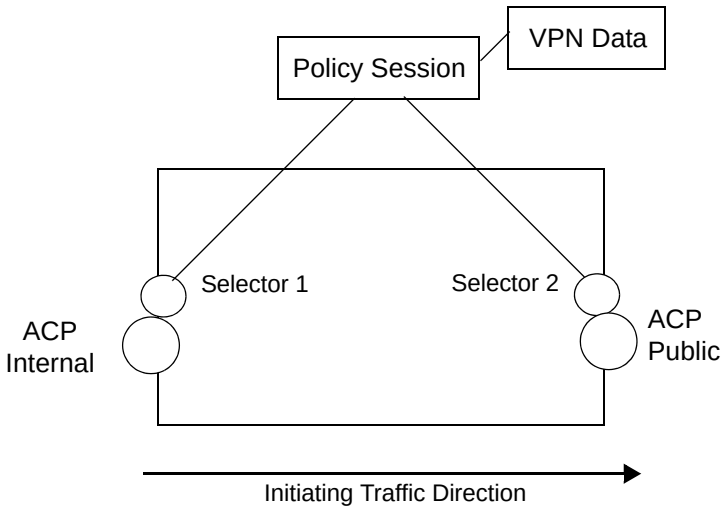


Figure 1. Architecture of a Policy Session

Figure 1 above shows a policy session that consists of two selectors, Selector 1 and Selector 2. Each selector is linked to an interface and the ACP applied to that interface. Selector 1 is associated with the original traffic attributes and Selector 2 is associated with the return traffic attributes. Selector 2 will be a mirror image of Selector 1.

Each selector contains the following information:

- The ingress ACP of the traffic flow
- The protocol number
- The source and destination IPv6 addresses of the flow
- Any protocol-specific information such as the source and destination port for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) traffic
- The identifier, type, and code for Internet Control Message Protocol (ICMP) echo packets.

Policy sessions are active as long as their configured lifetime has not expired, or the session has not been cleared manually. When the lifetime for the session has expired, the session is deleted.

Each policy session has an assigned policy timeout. Policy timeouts specify the amount of time that should pass before policy sessions are deleted due to inactivity (for example, packets are not passing through the policy session). Policy session timeouts are used to configure the lifetime for policy sessions and are applicable to the following four different policy session states:

1. **Pre-established TCP policy session**. This type of session is one that has been opened by TCP, on which the full three-way TCP handshake has not yet been observed.
2. **Established TCP policy session**. This type of session is one in which a three-way handshake has been observed, but the receipt of an RST from either endpoint or a FIN from both endpoints has not occurred. This type of session can also be a stateless policy session which has not received an RST from either endpoint or a FIN from both endpoints, or a policy sessions for all other protocols.
3. **TCP FIN policy session (closed session)**. This type of session is one in which a FIN has been received from both endpoints.
4. **TCP RST session (closed session)**. This type of session is one in which an RST has been received from either endpoint.

Each session state has a specific command used to govern the policy timeout. For established sessions, any traffic matching the policy session will refresh the lifetime for the session. If the policy session has no traffic during the session's lifetime, the session is closed. For pre-established or closed TCP sessions, the lifetime is set when the policy session enters the particular state. Traffic matching the policy session does not refresh the lifetime, and when the lifetime expires, the policy session is closed.

Creating Policy Sessions

There are several different circumstances that trigger the creation of a policy session:

1. A policy session is created when the first packet of a unique traffic flow arrives. Once the session is created, all other packets in the traffic flow will match the policy session's 5-tuple selector and new sessions do not need to be created.
2. A policy session is created when traffic arrives that matches an existing passive association.
3. Policy sessions are created for existing open local socket connections when the firewall is toggled from disabled to enabled. A local socket connection is a TCP connection, like Telnet or Secure Shell (SSH), to the unit itself. A policy session is created for this type of connection so that traffic over the open socket is not disconnected from the firewall.

Reworking Active Policy Sessions

When a policy session is reworked, one or more of the important values in the association structure are changed. A rework is triggered by either a route table change or a change to an ACL in a PBR entry. Only active policy sessions are reworked.

A true rework is the most common type of rework. The only values in the selector that are changed in this type of rework are the ACP and the interface with which that selector is associated. A true rework simply replaces the ACP and/or interface on an association's selector on one end of the session with a different ACP and/or interface.

Another type of rework is a virtual private network (VPN) data rework. *Figure 1 on page 3* shows that policy sessions include pointers to VPN data structures, which connect firewall associations with VPN security associations (SAs). When route table changes affect which crypto maps should be used by policy sessions, the policy sessions must be reworked so they use the correct VPN SAs.

Deleting Active Policy Sessions

Different events can lead to the deletion of an active policy session. Some of these events are triggered manually, while others are triggered by another protocol or due to inactivity.

All currently active policy sessions can be manually deleted using the command **clear ipv6 policy-sessions**. This command also contains optional parameters that allow the user to specify policy sessions limited to a particular VRF, only policy sessions related to a particular ACP, or a specific policy session. Refer to *Troubleshooting on page 47* for more information on the **clear ipv6 policy-sessions** command.

An active policy session will be deleted when the lifetime, initially set to the policy timeout, decrements to zero. Refer to *Active Policy Sessions on page 3* for a detailed discussion on policy timeouts.

When a VRF instance is deleted, then all active policy sessions using that VRF are also deleted. Another application where active policy sessions must be deleted is the transfer to a backup router when running Virtual Router Redundancy Protocol (VRRP). VRRP is a protocol where two or more physical routers advertise themselves as one virtual router, but only one of the physical routers performs the actual routing at any given time. If the currently active physical router in the group fails, then one of the other routers will immediately take over routing duties; thus increasing network connection reliability. However, when this transfer occurs, active policy sessions using the interface to which the failed physical router is connected must be deleted.

There can be up to a 7-second delay before the policy session is actually deleted. Also, when a TCP policy session is deleted, the AOS device will send TCP RSTs to both ends of the session (as long as the session is not stateless and if the unit has not already seen a FIN for the session).

Local Traffic Processing

The AOS IPv6 firewall can be configured to process local traffic only, that is, traffic arriving at the unit's local IP stack. When configured, routed traffic is allowed to flow through the AOS unit uninspected, but locally destined traffic is inspected by the firewall. This feature allows the firewall to protect local services running on the AOS unit even when routed traffic bypasses the firewall. When local traffic processing is enabled, several other security features are impacted, such as IPsec, policy classes, IP route cache, and Generic Routing Encapsulation (GRE).

- Local traffic only firewall processing cannot be used with cryptography (**ipv6 crypto**) because for IPsec to function, traffic must proceed through the firewall. If the firewall is configured to process local traffic only, routed traffic that requires IPsec protection will not flow through the firewall and therefore will not receive IPsec protection.
- Policy classes are applied only to traffic destined to the local stack when local traffic processing is enabled. The **self** policy class is applied to local traffic originating from the local stack, allowing all traffic, and cannot be changed.
- IP route cache entries are not created for local destinations or for the loopback interface when local traffic processing enabled.
- Local Generic Routing Encapsulation (GRE) traffic encapsulated by a GRE tunnel interface will bypass the firewall when local traffic processing is enabled.

Stateless Processing

Stateless processing refers to when a packet is processed without full consideration being given to what traffic the unit has already seen. In other words, it doesn't care about the current state of the traffic flow; the packet is considered apart from previous packets (except that attack checks might be performed). As mentioned earlier, access group processing is always stateless. The following sections describe other cases of stateless processing when the firewall is enabled.

Stateless Processing That Creates Policy Sessions

In some cases, statelessly processed traffic creates active policy sessions. The traffic is not susceptible to firewall timeouts and does not go through an application-level gateway (ALG) since ALGs perform stateful processing. However, depending on the protocol, stateless traffic might undergo attack checks.

The following methods are used to specify traffic that should not undergo ALG or stateful attack checks, but for which a policy session should be created:

- Enable **ipv6 crypto,** but leave **ipv6 firewall** disabled. In this case, the **ipv6 crypto** command allows VPN policy sessions and only applies the default and self ACP. (Traffic will be processed statelessly.)
- Add the keyword **stateless** to the end of an **allow** statement in an IPv6 ACP. Traffic that matches the **allow** statement will create a policy session, but will not undergo attack checking. This method is most commonly used to allow traffic between internal subnets that need to communicate with each other, for example, network 2001:DB8:1::x communicating with network 2001:DB8:2::x.

There are several types of traffic where it is helpful or necessary for a stateless policy session to be created:

- VPN traffic going to an interface with the default ACP, but **ipv6 firewall** is disabled.
- Routing Layer 4 protocol traffic that is not well known. This is traffic that is not TCP, UDP, ICMP, GRE, authentication header (AH), encapsulating security payload (ESP), internet group management protocol (IGMP), open shortest path first (OSPF), protocol independent multicast (PIM), or VRRP since this type of traffic must bypass attack checks or it will be dropped.
- Encrypted traffic not destined for the local router, that must be able to pass through the unit. This traffic could be dropped if it went through the firewall.
- Packets within a certain traffic flow that will use the same ACP as every other packet. A configuration in which the ingress and/or egress interfaces of different packets could be different (such as a remote unit load sharing the traffic back to the local router).

There is also a method for specifying traffic that should not go through an ALG, but should undergo attack checks and for which a policy session should be created:

- Disable specific ALGs that you do not want to use to process traffic.

Stateless Processing That Does Not Create Policy Sessions

For the following types of statelessly processed traffic, active policy sessions are not required because the traffic is implicitly allowed by AOS firewall:

- Dynamic Host Control Protocol (DHCP) traffic to or from the unit (not through the unit). When the unit is a DHCP server, this can mean broadcast packets sent by a DHCP client seeking an IPv4 address. When the unit has a DHCP client interface of its own, this can mean unicast packets sent to it by a DHCP server.

- Multicast traffic, including OSPF, PIM, IGMP, Routing Information Protocol version 2 (RIPv2), and VRRP traffic. (Other types of multicast traffic are unrecognized by AOS.)
- Internet key exchange (IKE) traffic (using UDP ports 500 and 4500) that terminates in the unit.
- ESP and AH traffic that terminates in the unit.

Application-Specific Processing

Certain applications need special handling to function correctly in the presence of a firewall. AOS uses ALGs for these applications. ALGs are aware of protocols not easily integrated with firewalls, and create a policy session that allows these protocols to work transparently. For example, the FTP ALG creates policy sessions that allow the control session (using TCP Port 21) to pass data and also creates policy sessions that allow the server-initiated data sessions to function (using TCP Port 20) in active FTP mode. This allows FTP clients to pass through the AOS firewall and ACPs without using FTP passive mode.

Several purposes for using ALGs include:

- Allowing application-specific deep packet inspection (DPI), meaning that it is not just the header of an incoming packet that is inspected but also the packet data. This information can help the ALG decide if the packet needs to be rerouted, modified, or dropped.
- Creating pending policy sessions for expected application-specific traffic, allowing the traffic to go through the unit without being dropped by the firewall.
- Increasing security by performing additional attack checks directly related to the type of traffic being handled by the ALG.

The AOS IPv6 firewall includes ALGs for handling the following applications and protocols:

- IPv6 File Transfer Protocol (FTP)

Refer to *Configuring the IPv6 FTP ALG for the IPv6 Firewall on page 22* for more information on enabling or disabling the ALGs.

Attack Checking

Through attack checking, the IPv6 firewall detects and discards traffic that matches profiles of known networking exploits or attacks. Using the **ipv6 firewall [vrf** <*name*>**]** command to enable firewall attack protection automatically detects and blocks these attacks. Some attack checks can be manually disabled, while other attack checks are enabled any time the firewall is enabled. Attack checks that can be enabled and disabled include: SYN flooding, TCP reset sequence number checking, reflexive traffic, and IP spoofing.

Table 1 on page 8 outlines the types of traffic discarded by the firewall. Since many attacks use similar invalid traffic patterns, attacks other than the examples listed in the table could also be blocked by the firewall. Also noted are whether the firewall process can be bypassed or the attack check disabled. For an additional list of packet checks performed by the firewall, that cannot be disabled, refer to *Table 5 on page 51*.

Table 1. Traffic Blocked by IPv6 Firewall Attack Protection Engine

Invalid Traffic Pattern	AOS IPv6 Firewall Response	Common Attacks	Disable or Bypass
Larger than allowed packets (packet size exceeds 65535)	Any packets that are larger than those defined by standards will be dropped.	Ping of Death	
Fragmented IP packets that produce errors when attempting to reassemble	The firewall intercepts all fragments for an IP packet and attempts to reassemble them before forwarding to the destination. If any problems or errors are found during reassembly, the fragments are dropped. Errors include less than minimum fragment size, tiny or overlapping fragments (RFC 1858), more than 114 fragments, last fragment length changed, and reassembly timeouts (set at 5 seconds).	SynDrop, TearDrop, OpenTear, Nestea, Targa, Newtear, Bonk, Boink	
Smurf Attack	The firewall drops any ICMP ping and UDP echo responses that are not part of an active session.	Smurf Attack	Bypassed in stateless processing
IP Spoofing	The firewall drops any packets with a source IPv6 address that appears to be spoofed. The IPv6 route table is used to determine if a path to the source address is known (out of the IP interface from which the packet was received). For example, if a packet with a source IPv6 address of 2001:DB8:1::1 is received on interface FR 1.16 and no route to 2001:DB8:1::1 (through interface FR 1.16) exists in the route table, the packet is dropped. Traffic that bypasses spoofing checks includes packets from the router itself, Dynamic Host Configuration Protocol version 6 (DHCPv6) traffic, multicast and routing protocol traffic, and Virtual Router Redundancy Protocol (VRRP) traffic. Spoofing detection can be turned off on individual ACPs to allow policy-based routing (PBR) or for any other case in which it would drop traffic that should not be dropped.	IP Spoofing	Can be disabled using the **rpf-check** parameter of the **ipv6 policy-class** command. In addition, DHCP traffic, allowed UDP multicast, and VRRP configurations can be used to enable spoof detection.
ICMP Control Message Floods and Attacks	The following types of ICMP packets are allowed through the firewall: echo, echo-reply, timestamp, timestamp reply, dest unreachable, quench, time exceeded, router solicitation and advertisement, neighbor solicitation and advertisement, multicast listener query, multicast listener report, multicast listener done, and multicast listener report v2. These ICMP messages are only allowed if they appear to be in response to a valid session. All others are discarded.	Twinge	

Table 1. Traffic Blocked by IPv6 Firewall Attack Protection Engine *(Continued)*

Invalid Traffic Pattern	AOS IPv6 Firewall Response	Common Attacks	Disable or Bypass
Falsified IP Header Attacks	The firewall verifies that the packet's actual length matches the length indicated in the IPv6 header. If it does not, the packet is dropped. These packets can include those where the IP header is less than the minimum, or the packet length is equal to or less than the header length (indicating no data in the packet).	Jolt/Jolt2	
Land Attack	Any packets with the same source and destination IPv6 addresses are discarded.	Land Attack	
Multicast Source IP	Packets with a multicast source IPv6 address are discarded.		
Invalid TCP Initiation Requests	Initial TCP SYN packets that have ACK, URG, RST, or FIN flags set are discarded.	Blind Spoof	Bypassed in stateless configuration.
Invalid TCP Segment Number RST	The sequence numbers for active TCP sessions are maintained in the firewall session database. If the firewall receives an RST segment with an unexpected (or invalid) sequence number, the packet is dropped. In addition, TCP data received after an RST is dropped.		Bypassed in stateless configuration. Can be disabled using **check tcp-seq-and-ack** parameter of **ipv6 firewall** command.
IP Options	All packets containing the IP source route option are dropped. In addition, packets with zero option length, option length missing, option length exceeds header length, strict or loose routing options, duplicate IP options, unknown IP options, or multiple Pad1 options are dropped.		Can be disabled using the **check duplicate-options**, **check unknown-options**, and **check multiple-pad1** parameters of the **ipv6 firewall** command.
Invalid IP Header Order	Packets with invalid IP header orders are discarded.		Can be disabled using the **check header-order** parameter of the **ipv6 firewall** command.
IP packet less than minimum fragment size	Packets with less than the minimum fragment size are dropped.		Can be disabled using the **check min-fragment-size** parameter of the **ipv6 firewall** command.
Invalid UDP Packet Lengths	UDP packets with lengths greater than the IP payload length or greater than the packet length minus the IP header length are discarded. In addition, UDP packets that are too short for the UDP header are discarded.	TARGA3	

Table 1. Traffic Blocked by IPv6 Firewall Attack Protection Engine *(Continued)*

Invalid Traffic Pattern	AOS IPv6 Firewall Response	Common Attacks	Disable or Bypass
Unsupported IP Protocol	Packets of unsupported IP protocols are discarded. Supported protocols include TCP, UDP, ICMP, GRE, AH, ESP, OSPF, PIM, IGMP, and VRRP.		Bypassed in stateless configuration.
Invalid TCP Header Length	TCP packets that are too short for the minimum length, incorrect length, or with header lengths shorter than the packet are discarded.		
TCP packets with no flags set and seq=0	These packets are discarded.	TCP Null Scan Attack	
UDP Packet Checksum Errors	UDP packets with checksum values of zero are discarded.		Can be disabled using the **check udp-checksum-zero** parameter of the **ipv6 firewall** command.
ICMP Packet Errors	ICMP packets with short headers or too small ICMP error date are discarded, as are ICMP packets with no source interface.		
Unidentifiable Route Source	If the firewall cannot detect a route for a source (prior to a spoofing check), the packet is discarded.		Can be disabled using the **rpf-check** parameter of the **ipv6 policy-class** command.
Proxy FTP packets targeting an IP service	When the FTP ALG is enabled, the firewall recognizes an attack as any extended port command (EPRT) sent by the FTP client with a port number less than 1024 and closes the connection.	Bounce, PORT Misdirection	Bypassed in stateless configuration. Can be disabled using the **check ftp-bounce** parameter of the **ipv6 firewall** command.

Static Filters

Static filters are able to filter IP traffic using access control lists (ACLs) without the firewall enabled. As a result, the attack checks that are normally performed by the firewall are not affected. A separate filtering decision is made for every individual packet without regard to the states of the previous packets. In this regard, static filters are stateless. The static filter will either allow or discard a packet.

There are two types of static filters used by AOS:

- Access groups - These are applied to an interface and contain filter lists that can be set to filter either inbound or outbound traffic. To filter both inbound and outbound traffic, you must apply two access groups to the interface. Even if the same access group is to be used for both directions, it must be applied twice. Access groups can be used to filter any type of traffic to, from, or through the unit. The advantages of using access groups is they do not require the firewall to be enabled.

- Access classes - These are applied to specific servers that reside in the unit, such as the Telnet or Hypertext Transfer Protocol (HTTP) server. They filter login access to the unit for the specific protocol associated with the server. Standard ACLs are used as the filter for access classes.

When access groups are used, an inbound access group applied to the public interface typically needs to reject sessions initiated from the Internet while allowing responses to sessions initiated from the internal network. This can be difficult to accomplish because the filter lists have no knowledge of the status of the session (sequence numbers, inactivity time, etc.). As a result, it is possible that an attacker could fool the configured filter lists and direct malicious traffic through the firewall.

For example, consider an application where a host located behind a firewall device initiates an outbound session to a server on the Internet. If access groups are used, two must be defined and applied to interfaces in the following manner:

- An access group that allows outbound traffic from the host to the Internet must be applied inbound on the internal interface.
- An access group that allows inbound traffic (responses to the initiated session) from the Internet to the host must be applied inbound on the public interface.

IPv6 Access Control Lists and Access Control Policies

IPv6 ACLs and ACPs regulate traffic through the routed network. When designing your traffic flow configuration, it is important to keep the following in mind:

- An ACL serves as a packet selector, defining exactly which packets should take the given action.
- An ACP defines the action to take on the packets selected by the ACL.
- An ACL is inactive until it is assigned to an active ACP.
- An ACP is inactive until it is assigned to an interface and the firewall is enabled.

Figure 2 on page 12 illustrates the steps necessary for activating IPv6 ACLs and ACPs.

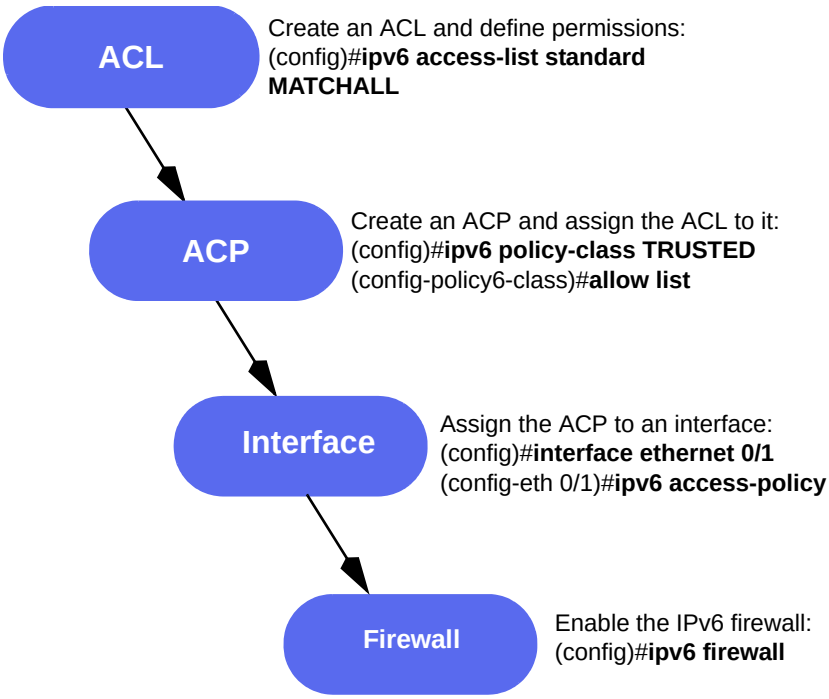


Figure 2. Activating IPv6 ACLs and ACPs

IPv6 Access Control Lists

IPv6 ACLs are used as packet selectors and are composed of an ordered list of entries. Each ACL entry begins with either a **permit** or **deny** keyword. A **permit** ACL is used to allow packets meeting the specified pattern to be processed by the feature using the ACL (in the case of the firewall, an ACP). A **deny** ACL causes packets meeting the specified pattern to advance to the next ACP entry.

When an ACL is being used to inspect an IP packet, the entries in the ACL are processed from top-to-bottom, in the order in which the entries were added to the ACL. If an IP packet does not match the criteria specified by a certain entry, then it is compared to the criteria in the next entry. If all of the entries in an ACL are examined and there is no match, there is an implicit **deny all** at the end of the ACL that causes the next rule in the ACP to be processed.

There are two types of ACLs, standard and extended.

- **Standard**
  The only field in the packet inspected by standard ACLs is the IPv6 source address. Packets sent from specific subnets or specific hosts can be specified as either **permit** or **deny**. The **any** keyword is used to specify either a **permit** or a **deny** of all packets. For standard ACLs (but not extended), an ACL cache is created to speed up the packet matching process.
- **Extended**
  There are a number of fields in the packet that can be inspected by extended ACLs. These include protocol, source and destination IPv6 addresses, source and destination ports, and most fields in the IP, TCP, UDP, and ICMP headers. The **any** keyword can be used in reference to the packet's source, destination, or both.

When entries exist within an ACL, an implicit deny statement is automatically placed at the end of the entries. When an ACL is empty (no entries), the ACL implicitly allows everything.

IPv6 Access Control Policies

IPv6 ACPs are applied when IPv6 packets are received on an inbound interface (rather than outbound). Every inbound interface can have only one associated ACP. They are faster to process than access groups. Each IPv6 ACP entry consists of a selector (an ACL) and an action (allow or discard). Also, an ACP entry can specify a destination ACP to match as part of its rule. When packets are received on an interface, the configured ACPs are applied to determine whether the data is processed or discarded. After inspecting traffic, the IPv6 ACP performs one of four actions:

- The **allow** keyword permits traffic through the firewall without changing the IP packet's source or destination IPv6 addresses or Layer 4 ports.
- The **discard** keyword drops the traffic.

ACPs are order dependent. When an IPv6 packet is evaluated, the matching engine begins with the first entry in the list and progresses through the entries until it finds a match. The first entry that matches is executed. Typically, the most specific entries should be at the top and the more general at the bottom.

There are several types of ACPs, including:

- The **self** ACP handles traffic whose source or destination is the router itself. This ACP is unique and cannot be deleted.
- The **default** ACP is applied to every interface that does not have a user-defined ACP and allows all traffic while performing stateful checks with **ipv6 firewall** enabled. It will allow all traffic while performing stateless checks if **ipv6 crypto** is enabled while **ipv6 firewall** is disabled. The default ACP is unique and cannot be deleted.
- The **user-defined** ACP is defined by the user. AOS supports up to 20 user-defined ACPs.

IPv6 ACP Entry Matching

As described previously, an IPv6 ACP entry can specify an **allow** or **discard** action. In order for an action to be taken on the IPv6 packet being inspected, it must first fit the match criteria. There are up to three steps taken when performing IPv6 packet matching on an ACP entry:

1. Check the optional destination ACP, if it exists. AOS reviews the IP packet’s destination IPv6 address and uses the route table to determine what the IP packet’s destination interface is for that address. If the ACP applied to that interface is the same as the ACP specified in the entry, the process continues to Step 2. Otherwise, the match fails and there is no need to perform Step 2.

It is recommended that this Step 1 above always be used on ACP entries. It can be useful if the same ACL is used on multiple ACP entries within the same ACP. If load sharing is being used, this will differentiate traffic that is being sent out two different interfaces, each of which will have a different associated ACP.

2. Check the ACL.

> **NOTE** *If no entries in an IPv6 ACP match the IP packet, then the packet is dropped by the last entry, which contains an implicit **discard all**. Also, if an IPv6 ACP has been created, but contains no entries, then the IPv6 packet is dropped.*

Processing ACPs and ACLs

The logical flow of how IP packets are processed by ACPs is described below, followed by ACLs. This information is provided to gain understanding using ACPs and ACLs together in an IPv6 firewall configuration.

ACPs

If the interface on which a packet arrives references an ACP that has not been created in the configuration, then the packet is dropped. The packet will also be dropped if the ACP has been created but contains no entries. If entries exist, they are processed from top-to-bottom, as they appear in the running configuration. If none of the entries match the packet, then the packet is dropped by the last entry, which contains an implicit **discard all**.

Exceptions

- Duplicate entries are allowed.
- IKE messages (using UDP ports 500 and 4500) to or from the unit are implicitly allowed unless they are matched by an explicit entry, in which case the action specified by the entry is taken.
- ESP messages to or from the unit are also implicitly allowed.
- Decrypted traffic must be explicitly allowed.

ACLs

If an ACP entry references an ACL that has not been created in the configuration, then the packet is matched by the entry unless a destination ACP was also specified and has already denied the packet. If the referenced ACL exists but contains no entries (an empty ACL), then the packet is matched. If entries exist, entries are processed from top-to-bottom, which is the order in which they were entered.

Exceptions

- Duplicate entries are NOT allowed.
- The last entry is an implicit **deny any**, meaning the packet is not matched by the ACL.

Hardware and Software Requirements and Limitations

Firewall protection is available on the AOS data and voice products as outlined in the *Product Feature Matrix* available online at https://supportforums.adtran.com. Some commands are interface specific and may not be available on all platforms listed. Use the online help (**?** in the CLI) to determine if a command is supported by your hardware platform.

In AOS firmware release R11.4.0, support for local firewall processing was implemented.

In AOS firmware release R11.10.2, the ability to match or block non-initial fragments in ACL entries was implemented.

For more detailed information about IPv6 in AOS, IPv6 ACL and ACP configuration, and other IPv6 features, refer to the configuration guide *IPv6 in AOS*, available online at https://supportforums.adtran.com.

Configuring the IPv6 Firewall

To configure the IPv6 firewall, you will need to enable the firewall and configure it's parameters. The IPv6 firewall configuration is outlined in the following sections:

1. *Enabling the IPv6 Firewall and Its Security Parameters on page 15*
2. *Configuring IPv6 Firewall Policy Timeouts on page 18*
3. *Configuring the IPv6 FTP ALG for the IPv6 Firewall on page 22*
4. *Configuring IPv6 Firewall Filtering Behavior on page 24*

In addition to configuring the IPv6 firewall, you will also need to configure any IPv6 ACLs or ACPs necessary for your network. IPv6 ACL and ACP configuration is outlined in the following sections:

1. *IPv6 ACLs on page 27*
2. *IPv6 Access Control Policies on page 33*
3. *Applying IPv6 ACLs and ACPs to an Interface on page 36*

Enabling the IPv6 Firewall and Its Security Parameters

The following commands are IPv6 commands used to configure the IPv6 firewall and security features in the AOS product.

1. Enable AOS IPv6 security features, including the stateful inspection firewall and IPv6 ACPs, using the **ipv6 firewall [vrf** <*name*>**] [local-traffic-only]** command. Packets will not be processed by ACPs until this command has been issued. Using the **no** form of this command disables the IPv6 security functionality. When the firewall is disabled, no security data processing is attempted. The optional **vrf** <*name*> parameter enables or disables the firewall for a nondefault (named) VRF. By default, the IPv6 firewall is disabled. The optional **local-traffic-only** parameter enables the firewall for processing local traffic only. Forwarded traffic is not sent to the firewall when this option is enabled. To enable the firewall, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall**

2. Specify the number of possible attack conditions the firewall will identify and block before generating a log message using the **ipv6 firewall [vrf** <*name*>**] attack-log threshold** <*number*> command. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. The <*number*> parameter specifies the number of possible attack conditions AOS identifies and blocks before generating the log. Valid range is **1** to **4294967295**. By default, a log message is generated after **100** possible attack conditions have been identified and blocked.

To change this number, enter the command from the Global Configuration mode prompt as follows:

(config)#**ipv6 firewall attack-log threshold 150**

3. Specify the number of policy events AOS will identify before generating a log message using the **ipv6 firewall [vrf** <*name*>**] policy-log threshold** <*number*> command. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. The <*number*> parameter specifies the number of policy events AOS identifies before generating the log. Valid range is **1** to **4294967295**. By default, a log is generated after **100** policy events have been identified. To change this number, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall policy-log threshold 150**

4. Specify the IPv6 firewall operates in stealth mode using the **ipv6 firewall [vrf** <*name*>**] stealth** command. The **no** form of this command disables stealth mode. By default, stealth mode is disabled. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. To enable stealth mode on the default VRF, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall stealth**

5. Reflexive traffic refers to packets that are routed out of the same interface on which they arrived. Enable stateful inspection of reflexive traffic by using the **ipv6 firewall [vrf** <*name*>**] check reflexive-traffic** command. By default, reflexive traffic is allowed to bypass the firewall and does not create a policy session. Using the **no** form of this command disables reflexive traffic checking, which is also disabled by default. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. To enable reflexive traffic checking, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall check reflexive-traffic**

6. Use the **ipv6 firewall [vrf** <*name*>**] check duplicate-options** command to specify that packets with duplicate options within a Destination Options or Hop-by-Hop Options extension header are dropped. The **no** form of this command allows packets with duplicate options. By default, packets containing duplicate options are dropped. The optional **vrf** <*name*> parameter specifies a nondefault VRF on which to drop duplicate option packets. If no VRF is specified, duplicate option packets on the default VRF will be dropped. To specify that no packets with duplicate options are forwarded, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall check duplicate-options**

7. Use the **ipv6 firewall [vrf** <*name*>**] check header-order** command to enable the dropping of packets that have extension headers in an order different than the recommendations proposed in the Systems and Network Analysis Center (SNAC) document *Firewall Design Considerations for IPv6*. By default, this check is enabled. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. Using the **no** form of this command disables the feature and allows the processing of packets with extension headers according to RFC 2460. To enable this check, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall check header-order**

8. Use the **ipv6 firewall [vrf** <*name*>**] check min-fragment-size** <*value*> command to specify the smallest permitted size for fragmented packets. Packets less than the specified size are dropped. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. The <*value*> parameter specifies the packet size in octets. Valid range is **56** to **1280**. By default, the minimum fragmented packet size is set to **640** octets. Using the **no** form of this command resets the minimum fragmented packet size to the default value. To change the smallest permitted size for fragmented packets, enter the command from the Global Configuration mode prompt as follows:

   (config)#**ipv6 firewall check min-fragment-size 800**

9. Use the **ipv6 firewall [vrf** <*name*>**] check multiple-pad1** command to specify that packets with more than one Pad1 option back-to-back within the Destination or Hop-by-Hop Options extension headers are dropped. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. By default, this check is disabled. Use the **no** form of this command to allow packets with more than one Pad1 option back-to-back. To enable the dropping of packets with more than one Pad1 option, enter the command as follows:

   (config)#**ipv6 firewall check multiple-pad1**

10. Use the **ipv6 firewall [vrf** <*name*>**] check udp-checksum-zero** command to specify that UDP packets with a checksum value of zero are dropped. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. By default, this feature is enabled. Using the **no** form of this command allows UDP packets with a checksum value of zero. To specify that UDP packets with a checksum value of zero are allowed, enter the command from the Global Configuration mode prompt as follows:

    (config)#**ipv6 firewall check udp-checksum-zero**

11. Use the **ipv6 firewall [vrf** <*name*>**] check unknown-options** command to specify that packets with unknown options in a Destination or Hop-by-Hop Options extension headers are dropped. The optional **vrf** <*name*> parameter identifies the firewall instance being configured. An instance of the IPv6 firewall exists for each configured VRF. If no VRF is specified, the default VRF is configured. By default, this check is enabled. Using the **no** form of this command allows packets with unknown options. Supported options include the following:
    - Pad1 (0)
    - PadN (1)
    - Tunnel encapsulation limit (4, Destination Options extension header only)
    - Router alert (5, Hop-by-hop Options extension header only)
    - Quick start (6)
    - CALIPSO (7)

    To enable the dropping of packets with unknown options, enter the command as follows:

    (config)#**ipv6 firewall check unknown-options**

Checking for Valid TCP Sequence Numbers and ACK Numbers in the IPv6 Firewall

When the TCP sequence and ACK number check is enabled, the IPv6 firewall inspects each packet of a TCP flow to ensure that the sequence numbers and ACK numbers are within the expected window for that session.

The TCP sequence and ACK number check is enabled by default on all VRF instances (whether default or named). To enable the sequence and ACK number check, enter the **ipv6 firewall [vrf** <*name*>**] check tcp-seq-and-ack** command from the Global Configuration mode. The optional **vrf** <*name*> parameter of this command specifies a nondefault (named) VRF instance on which to enable or disable the sequence and ACK number check. If no VRF instance is specified, the action is performed on the default VRF instance. To enable the TCP sequence and ACK number check on the nondefault VRF instance **Red1**, enter the command as follows:

(config)#**ipv6 firewall vrf Red1 check tcp-seq-and-ack**

Configuring IPv6 Firewall Policy Timeouts

When the IPv6 firewall allows traffic to pass through the router (based on the ACP configuration), it creates a policy session to identify that particular flow of traffic. Having the policy session allows faster traffic flow through the firewall because the policy session contains traffic flow information that uniquely identifies that flow. This information includes: the ingress ACP of the flow, the protocol number, the source and destination IPv6 addresses of the flow, and protocol-specific information such as the source and destination port for TCP and UDP, and the identifier, type, and code for ICMP echo packets. Policy sessions are maintained as long as their lifetime has not expired. When the lifetime for the session has expired, the session is deleted.

Policy session timeouts are used to configure the lifetime for policy sessions. Policy session timeouts are applicable to the following four different policy session states:

1. **Pre-established TCP policy session**. This type of session is one that has been opened by a TCP SYN, on which the full three-way TCP handshake has not yet been observed.
2. **Established TCP policy session**. This type of session is one in which a three-way handshake has been observed, but the receipt of an RST from either endpoint or a FIN from both endpoints has not occurred. This type of session can also be a stateless policy session which has not received an RST from either endpoint or a FIN from both endpoints, or a policy sessions for all other protocols.
3. **TCP FIN policy session (closed session)**. This type of session is one in which a FIN has been received from both endpoints.
4. **TCP RST session (closed session)**. This type of session is one in which an RST has been received from either endpoint.

Each session state has a specific command used to govern the policy timeout. For established sessions, any traffic matching the policy session will refresh the lifetime for the session. If the policy session has no traffic during the session's lifetime, the session is closed. For pre-established or closed TCP sessions, the lifetime is set when the policy session enters the particular state. Traffic matching the policy session does not refresh the lifetime, and when the lifetime expires, the policy session is closed.

Configuring Pre-established, TCP FIN, and TCP RST Policy Session Timeouts

To change the policy session for a specific policy session state (excluding established sessions), use one of the following commands:

1. To configure the policy session timeout for a pre-established TCP policy session, enter the **ipv6 firewall [vrf** <*name*>**] tcp-unestab-timeout** <*timeout*> command from the Global Configuration mode. The optional **vrf** <*name*> parameter selects a nondefault VRF. If this option is not specified, the action is performed on the default (unnamed) VRF. The <*timeout*> parameter specifies the timeout in seconds. The valid timeout range is **1** to **4294967295** seconds. By default, the timeout is set to **20** seconds. Using this command sets the time period allowed for TCP to complete the three-way handshake at the beginning of the connection. The **no** form of this command returns the timeout to the default setting. To change this type of policy session timeout, enter the command as follows:

   (config)#**ipv6 firewall tcp-unestab-timeout 30**

2. To configure the policy session timeout for a policy session closed by a bidirectional TCP FIN, enter the **ipv6 firewall [vrf** <*name*>**] fin-timeout** <*timeout*> command from the Global Configuration mode. The optional **vrf** <*name*> parameter selects a nondefault VRF. If this option is not specified, the action is performed on the default (unnamed) VRF. The <*timeout*> parameter specifies the timeout in seconds. The valid timeout range is **0** to **4294967295** seconds. By default, the timeout is set to **4** seconds. Using this command sets the time period allowed for the TCP FIN process. If the timeout is defined to be **0**, the policy session is deleted immediately without entering a post-connection state. This may be necessary for hosts that do not implement the TIME_WAIT TCP state correctly, but instead permit immediately reopening closed sessions. The **no** form of this command returns the timeout to the default setting. To change this type of policy session timeout, enter the command as follows:

   (config)#**ipv6 firewall fin-timeout 30**

3. To configure the policy session timeout for a policy session closed by a TCP RST, enter the **ipv6 firewall [vrf** <*name*>**] rst-timeout** <*timeout*> command from the Global Configuration mode. The optional **vrf** <*name*> parameter selects a nondefault VRF. If this option is not specified, the action is performed on the default (unnamed) VRF. The <*timeout*> parameter specifies the timeout in seconds. The valid timeout range is **0** to **4294967295** seconds. By default, the timeout is set to **20** seconds. Using this command sets the time period allowed for the TCP reset process. If the timeout is defined to be **0**, the policy session is deleted immediately without entering a post-connection state. This may be necessary for hosts that do not implement the TIME_WAIT TCP state correctly, but instead permit immediately reopening closed sessions. The **no** form of this command returns the timeout to the default setting. To change this type of policy session timeout, enter the command as follows:

   (config)#**ipv6 firewall rst-timeout 30**

Configuring Established Policy Session Timeouts

Established policy session timeouts are configured somewhat differently than timeouts for other policy session states. The commands used for established policy session timeouts customize timeout intervals for protocols (by specifying the protocol or a specific ACL), specific services (by specifying the port used or a specific ACL), and specific ingress ACPs. Multiple commands can be used to specify different timeouts for different protocols, services, and ingress ACPs. Using the **no** form of these commands returns the

timeout to the default value. The default values are **600** seconds for established TCP policy sessions, and **60** seconds for all other protocols. All policy session timeout commands are entered from the Global Configuration mode. The following commands are available to configure the policy session timeout for established policy sessions:

1. **ipv6 policy-timeout [vrf** <*name*>**] match** <*ipv6 acl name*> **[policy** <*ipv6 acp name*>**]** <*timeout*>
2. **ipv6 policy-timeout [vrf** <*name*>**] [all-protocols | ahp | esp | gre | icmpv6 | tcp | udp |** <*protocol number*>**] [policy** <*ipv6 acp name*>**]** <*timeout*>
3. **ipv6 policy-timeout [vrf** <*name*>**] [tcp | udp] [all-ports | range** <*beginning port*> <*ending port*> **|** <*port*>**] [policy** <*ipv6 acp name*>**]** <*timeout*>

The optional **vrf** <*name*> parameter selects a nondefault VRF. If this option is not specified, the action is performed on the default (unnamed) VRF.

The **match** <*ipv6 acl name*> parameter specifies that if the packet creating the policy session matches the specified ACL, the policy timeout specified by this command is used. Because an ACL can be used to specify protocol and port information, you do not need to specify port and protocol parameters when using this version of the command; however, you must define appropriate ACL entries that characterize the traffic that will use this particular policy timeout. If an ACL does not exist at the time you enter this command, an implicit ACL is created.

The optional **policy** <*ipv6 acp name*> parameter specifies that if the policy session uses the specified ACP as its ingress policy class, the policy timeout specified by this command is used (provided the ACL or protocol/port information matches, if specified). If the policy class does not exist at the time the command is issued, an implicit policy class is created.

The **all-protocols** parameter specifies a timeout for all protocols. This policy timeout is used if a more specific match is not found.

The <*protocol number*> parameter specifies the policy timeout for a particular numbered protocol. Valid protocol number range is **0** to **255**. The following are accepted protocol keywords and their associated numbers: **ahp** (51), **esp** (50), **gre** (47), **icmpv6** (58), **tcp** (6), and **udp** (17). Protocol numbers reserved for extension headers cannot be used, for example, you cannot use **0** (hop-by-hop options), **43** (routing), **44** (fragment), **59** (no next header), **60** (destination options), or **135** (mobility).

The **all-ports** parameter defines the policy timeout to be used for all ports of the specified protocol (TCP or UDP only) if a more specific match is not found.

The **range** <*beginning port*> <*ending port*> parameter allows a range of continuous ports to be specified (TCP or UDP only). Each parameter can range from **0** to **65535**.

The <*port*> parameter allows a single port to be specified (TCP and UDP only). Keywords are available for well-known ports, and are identical to the port list used for TCP and UDP extended ACLs (refer to *IPv6 ACLs on page 27*). Valid port range is **0** to **65535**.

The <*timeout*> parameter specifies the timeout in seconds for the policy session. Valid range is **1** to **4294967295** seconds. The default values are **600** seconds for established TCP policy sessions, and **60** seconds for all other protocols.

For example, to configure multiple policy session timeouts based on different protocols (and the associated ACLs), enter the commands as follows:

(config)#**ipv6 policy-timeout match A1 policy P1 1000**
(config)#**ipv6 policy-timeout match A2 policy P1 2000**
(config)#**ipv6 policy-timeout tcp ssh policy P1 3000**
(config)#**ipv6 policy-timeout tcp range 100 200 policy P1 4000**
(config)#**ipv6 policy-timeout tcp range 150 250 policy P1 5000**
(config)#**ipv6 policy-timeout tcp all-ports policy P1 6000**
(config)#**ipv6 policy-timeout match A3 7000**
(config)#**ipv6 policy-timeout tcp ssh 8000**

(config)#**ipv6 access-list extended A1**
(config-ex6-nacl)#**permit gre host 2001:DB8:1234:1::1 any**
(config-ex6-nacl)#**permit tcp any any eq www**

(config)#**ipv6 access-list extended A2**
(config-ex6-nacl)#**permit tcp host 2001:DB8:1234:1::1 any**

(config)#**ipv6 access-list extended A3**
(config-ex6-nacl)#**permit esp any any**

Application of the Policy Session Timeout

Because multiple policy session timeouts can be configured and applied to multiple protocols, ACLs, and policy sessions, the IPv6 firewall uses matching criteria to determine which policy session timeout should be applied to the session. When the IPv6 firewall creates a policy session, it will determine the correct policy timeout to apply using the following criteria in this order:

1. **Ingress Policy Class**. Policy timeouts whose specified ACP matches the ingress ACP of the newly created policy session are considered first, and then policy timeouts that do not specify a ACP.
2. **ACL**. Policy timeouts specifying an ACL are searched before policy timeouts that specify protocol numbers. Policy timeouts specifying an ACL are searched for a match in the order in which the policy timeouts were configured, beginning with the oldest policy timeout and proceeding to the most recently configured.
3. **Protocol Number**. If no policy timeouts with a matching ACL are found, the firewall looks next at the protocol number of the newly created policy session. If no match for a protocol number is found, then one of the following occurs:
   - The **all-protocols** timeout value (if defined) is selected.
   - The search resumes with policy timeouts that do not specify an ingress ACP.
   - The default **all-protocols** timeout is selected.
4. **Port**. If the protocol number is TCP or UDP, policy timeouts specifying ports are considered before policy timeouts that do not specify ports. Policy timeouts specifying a single port are searched before policy timeouts specifying a range of ports. Policy timeouts specifying a range of ports are searched for a match in the order in which the timeouts were configured, beginning with the oldest policy timeout first. If no policy timeout specifying a single port or a range of ports is matched, one of the following occurs:

- The **all-ports** timeout value (if defined) is selected
- The search resumes with policy timeouts that do not specify an ingress ACP.
- The default **all-ports** timeout is selected.

Configuring the IPv6 FTP ALG for the IPv6 Firewall

Certain applications send traffic using dynamic ports indicated during application signaling. ALGs are used to monitor such applications and dynamically allow traffic that would otherwise be discarded due to configured ACP rules.

Two purposes for using ALGs are:

- Creating pending policy sessions for expected application-specific traffic, allowing the traffic to go through the unit without being dropped by the firewall.
- Increasing security by performing additional attack checks directly related to the application being handled by the ALG.

The AOS IPv6 firewall includes ALGs for handling the following applications and protocols:

- IPv6 FTP

How the IPv6 FTP ALG Functions

The IPv6 FTP ALG operates by parsing the Layer 5 contents of packets used for FTP, and when necessary, opens pending policy sessions so FTP data transfers are able to traverse the IPv6 firewall without being dropped by configured ACPs. In addition, the IPv6 FTP ALG has the ability to perform FTP-specific attack checking.

During the processing of an FTP flow, the IPv6 FTP ALG creates a pending policy session based on a currently active policy session. This pending policy session listens for expected FTP data transfer traffic. These pending policy sessions can be viewed and cleared by using the commands as outlined in *Troubleshooting on page 47*.

Any IPv6 firewall policy sessions created using a stateless ACP entry bypass all ALG processing, even if the ALG is enabled for the ACP's destination port. This allows ALG processing to be enabled globally for a specific port or ports, but the global configuration to be bypassed under specific circumstances (such as on a particular ACP, or for particular hosts or networks based on IPv6 ACLs).

The IPv6 FTP ALG cannot be enabled on a protocol and port that is the default protocol and port for any other ALG, even if the other ALG is disabled. The IPv6 FTP ALG also cannot be enabled on a TCP port whose default filtering behavior has been overridden (refer to *Configuring IPv6 Firewall Filtering Behavior on page 24*).

Enabling the FTP ALG for the IPv6 Firewall

Instances of the FTP ALG can be enabled or disabled on any number of TCP ports, depending on specific network requirements. By default, the ALG for FTP is enabled on all VRF instances on TCP port 21. The ALG is disabled by default on any other port.

Disabling an ALG could cause the firewall to block some of the traffic for the specified application, depending on any configured ACP rules.

To enable the IPv6 FTP ALG, use the **ipv6 firewall [vrf** <*name*>**] alg ftp [tcp [port** <*port*>**]]** command from the Global Configuration mode. The optional **vrf** <*name*> parameter specifies a nondefault (named) VRF instance on which to enable the ALG. If no VRF instance is specified, the ALG is enabled on the default (unnamed) VRF instance. The optional **tcp** parameter specifies that the port which follows (if provided) is a TCP port. The optional **port** <*port*> parameter of the command allows you to specify a single port on which to enable or disable the IPv6 FTP ALG. The port specified corresponds to the destination port of the firewall policy session, which is the destination port of the initial traffic. Valid port range is **0** to **65535**. By default, the IPv6 FTP ALG is enabled on TCP port **21**. Using the **no** form of this command disables the FTP ALG for the specified port. Enter the command as follows to enable the IPv6 FTP ALG on the default VRF and on the default port (TCP port **21**):

```
(config)#ipv6 firewall alg ftp
(config)#
```

The following example disables the IPv6 FTP ALG on the default port (**21**), and then enables it on TCP ports **10000** and **20000**:

```
(config)#no ipv6 firewall alg ftp tcp port 21
(config)#ipv6 firewall alg ftp tcp port 10000
(config)#ipv6 firewall alg ftp tcp port 20000
```

Using the IPv6 FTP ALG to Protect Against FTP Bounce Attack

In addition to allowing the flow of IPv6 FTP traffic through the IPv6 firewall, the IPv6 FTP ALG can be used to protect against FTP bounce attacks. An FTP bounce attack is a network attack where malicious hosts using proxy FTP can target a specific well-known service on one server (Server A) by instructing another FTP server (Server B) to send a file to Server A that contains commands relevant to the service being attacked. For example, this can allow a malicious host to forge mail on Server A without making a direct connection. The lack of a direct file transfer between the attacker and the target server makes the identity of the attacker difficult to determine.

The IPv6 FTP ALG, however, can be used to protect against such an attack. When this feature is enabled, the IPv6 FTP ALG recognizes as an attack any EPRT sent by the FTP client that has a TCP port number less than **1024**, and the ALG closes the connection. The ALG performs this action because TCP port numbers in the range from **0** to **1023** are used by well-known services.

Although the IPv6 FTP ALG can perform bounce attack checks when ports less than 1024 are specified in an EPRT, services running on ports greater than 1023 are still vulnerable to FTP bounce attacks.

The FTP bounce attack check is enabled by default on all VRF instances (whether default or named). To enable the attack check, enter the **ipv6 firewall [vrf** <*name*>**] check ftp-bounce** command from the Global Configuration mode. The **no** form of this command disables FTP bounce attack check. The optional **vrf** <*name*> parameter of this command specifies a nondefault VRF instance on which to enable or disable bounce attack check. If no VRF instance is specified, the action is performed on the default (unnamed) VRF instance. To enable FTP bounce attack check on the nondefault VRF instance **RED1**, enter the command as follows:

(config)#**ipv6 firewall vrf RED1 check ftp-bounce**

Configuring IPv6 Firewall Filtering Behavior

In AOS firmware release R10.1.0, the ability to configure IPv6 firewall filtering behavior was introduced. Modifying the filtering behavior settings for a particular port and protocol allows ALG-like behavior in certain applications without requiring parsing of the Layer 5 packet contents used by these applications. By using this feature, you can enable firewall traversal for applications by adding one or more configuration options if no ALG is available for the application. A common application that can be handled by this feature is Trivial File Transfer Protocol (TFTP).

The ordinary filtering behavior of the IPv6 firewall is to restrict permitted return traffic to the exact source and destination IP addresses and ports of the initial traffic flow. This is called address- and port-dependent filtering. In some applications, including TFTP, the external traffic generated as part of the application can respond from a different external port than the one specified in the firewall configuration. This traffic might not be allowed to traverse the firewall, depending on the configured ACP rules. If available, an ALG could be used to accommodate such an application. The ALG would parse the application layer payload for traffic from the initiating host, and create an appropriate pending policy session to allow the expected response. With the release of R10.1.0, the IPv6 firewall incorporates two additional configurable filtering behaviors that can take the place of such ALGs for certain applications.

The first additional method of firewall filtering is address-dependent filtering. In this type of filtering, return traffic from an external host to the initiating internal host is allowed from any port, but traffic originating from any other external host will continue to be blocked. The second additional method of firewall filtering is endpoint-independent filtering. In this type of filtering, any external host can respond to traffic from the initiating internal host from any port.

For all firewall filtering behaviors, the return traffic from the external source must be destined to the original source address and port of the initial traffic. Traffic destined to some other port on the internal host will be blocked for all filtering behaviors.

Policy sessions created using a stateless ACP entry always use address and port-dependent filtering, and ignore any configured filtering behavior. Therefore, if address-dependent or endpoint-independent filtering is required for an application to successfully traverse the firewall (such as with TFTP), using a stateless policy class entry might result in limited or no connectivity for the application.

In addition, ALGs and address-dependent or endpoint-independent filtering behavior are mutually exclusive. They cannot be configured on the same port.

By default, most ports are filtered by traditional firewall filtering (**address-port-dependent**). UDP port 69, the TFTP port, uses **address-dependent** filtering by default.

Configuring IPv6 Firewall Filtering Behavior

The following illustrations depict the three types of filtering behavior available. Each illustration shows which return traffic is allowed back through the firewall, and which return traffic is blocked for each type of filtering behavior.

Figure 3 describes address- and port-dependent filtering behavior, which is the typical filtering behavior of the IPv6 firewall. In this scenario, traffic that is initiated from internal host **X** on port **x** that is destined to external host **Y** on port **y** results in the firewall only allowing external traffic that is the exact flow **Y:y** to **X:x**. Traffic that originates from a different source port on the external host **Y** to internal host **X** on port **x** is dropped. Traffic originating from any other external host **Z** to host **X** on port **x** is also dropped.

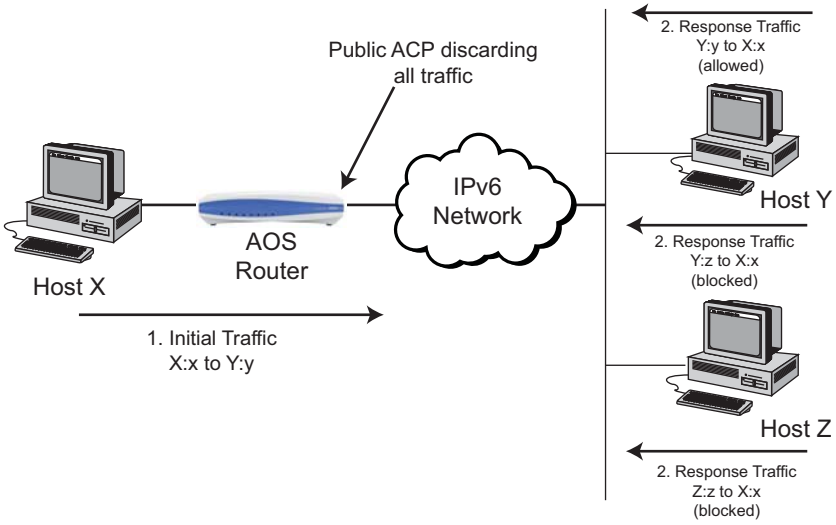


Figure 3. Address- and Port-Dependent Filtering Behavior

Figure 4 describes address-dependent filtering behavior. In this scenario, traffic that is initiated from the internal host **X** on port **x** and is destined to an external host **Y** on port **y** results in any return flow from host **Y** being allowed to reach the internal host **X** on port **x**. For example, this filtering behavior allows traffic from **Y:y** to reach **X:x**, as well as traffic from **Y:z** to reach **X:x**. Traffic originating from any other external host **Z**, however, continues to be blocked.

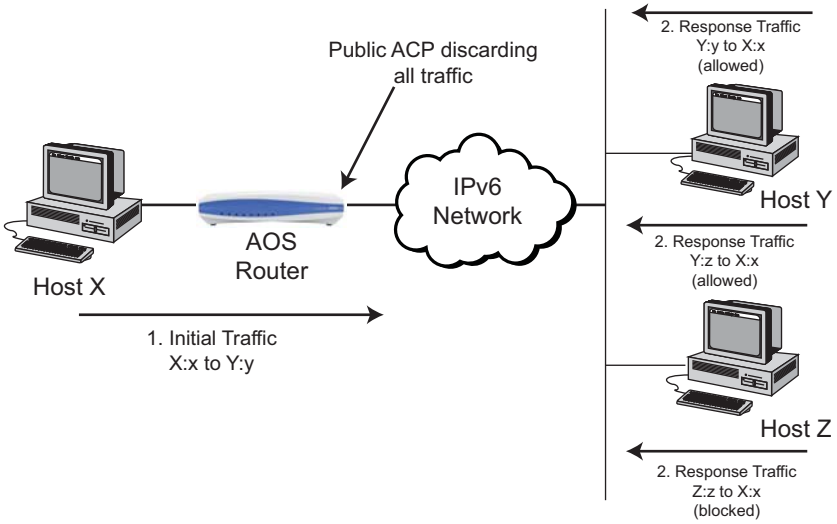


Figure 4. Address-Dependent Filtering Behavior

Figure 5 describes endpoint-independent filtering behavior. In this scenario, traffic that is initiated from the internal host **X** on port **x** that is destined to an external host results in any traffic from any external host, sourced from any port, being allowed to reach the internal host **X** on port **x**. For example, this filtering behavior allows traffic from **Y:y** to reach **X:x**, traffic from **Y:z** to reach **X:x**, and traffic from **Z:z** to reach **X:x**. This is the most lenient of all firewall filtering behavior types.

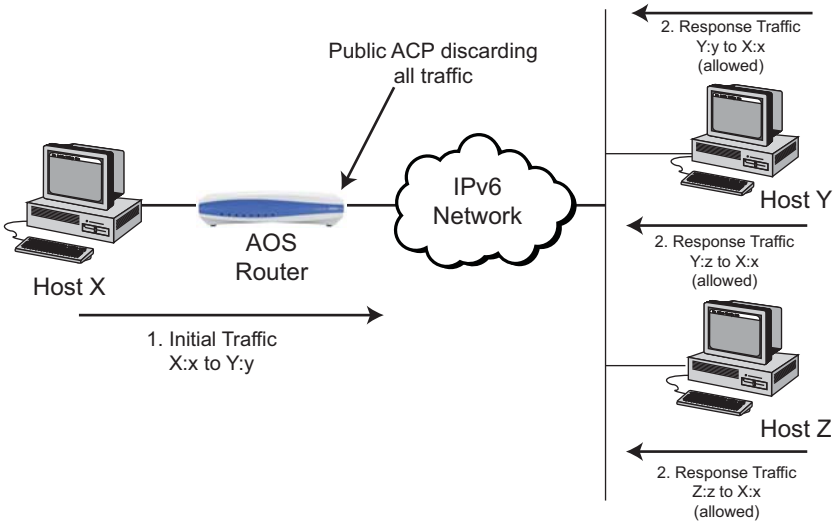


Figure 5. Endpoint-Independent Filtering Behavior

IPv6 firewall filtering behavior is modified using the **ipv6 firewall [vrf** <*name*>**] filtering-behavior [tcp** <*port*> **| udp** <*port*>**] [address-dependent | address-port-dependent | endpoint-independent]** command from the Global Configuration mode. The optional **vrf** <*name*> parameter specifies a nondefault VRF instance on which to modify the IPv6 firewall filtering behavior. If no VRF instance is specified, the filtering behavior for the default (unnamed) VRF instance is modified. The **tcp** <*port*> and **udp** <*port*> parameters specify either a TCP or UDP port for which the filtering behavior is changed. The port specified corresponds to the destination port of the firewall policy session, which is the destination port of the internal traffic. You can enter a keyword for a well-known port or a port number. Valid range is **0** to **65535**. The **address-dependent** parameter specifies that address-dependent filtering is used for traffic initiated using the specified destination port. The **address-port-dependent** parameter specifies that address- and port-dependent filtering is used for traffic initiated using the specified destination port. This is the default firewall filtering behavior for most ports. The **endpoint-independent** parameter specifies that endpoint-independent filtering is used for traffic initiated using the specified destination port.

For example, to specify that TCP port **10000** and UDP port **40** are filtered by endpoint-independent filtering and that TCP port **20000** and UDP port **30** are filtered by address-dependent filtering, enter the command as follows:

```
(config)#ipv6 firewall filtering-behavior tcp 10000 endpoint-independent
(config)#ipv6 firewall filtering-behavior tcp 20000 address-dependent
(config)#ipv6 firewall filtering-behavior udp 30 address-dependent
(config)#ipv6 firewall filtering-behavior udp 40 endpoint-independent
```

Configuring IPv6 ACLs and ACPs in AOS

IPv6 traffic control can be achieved in AOS by using ACLs, ACPs, and by specifying IPv6 firewall parameters. ACL and ACP configurations are discussed in the following sections.

IPv6 ACLs

The basic function of ACLs remains the same between IPv4 and IPv6. Packets either match a **permit** or a **deny** entry in the ACL. If no match is found based on the match criteria (referred to as a *miss*), the feature using the ACL must determine how to handle processing. For example, with typical IPv6 traffic, access groups treat a miss as a **deny**, effectually dropping the traffic. For IPv6 Neighbor Discovery (ND) messages, however, access groups treat a miss as a **permit**, effectually allowing rather than dropping the traffic.

As with IPv4, there are two ACL types in IPv6: standard ACLs and extended ACLs. The standard ACLs typically represent the source address of the packet and extended ACLs fully specify the source and destination components of a packet, as well as additional upper-layer matching parameters.

Creating Standard IPv6 ACLs

To create a standard IPv6 ACL, enter the **ipv6 access-list standard** <*ipv6 acl name*> command from the Global Configuration mode. This command creates and names the ACL, and enters the ACL's configuration mode. Using the **no** form of this command removes the ACL from the unit's configuration. For example, to create the standard IPv6 ACL **MATCHALL**, enter the command as follows:

```
(config)#ipv6 access-list standard MATCHALL
(config-std6-nacl)#
```


> **NOTE** *If you are using both IPv4 and IPv6 ACLs, they must have different names. You cannot assign an IPv4 ACL and an IPv6 ACL the same name or you will receive an error message.*

Once you have created the IPv6 ACL and entered the ACL's configuration mode, you can specify the match criteria used by the ACL. These criteria are order-sensitive, meaning criteria are processed in the order in which they were entered. Unlike IPv4, standard IPv6 ACLs do not support log entries. The following commands specify the match criteria for the standard ACL:

1. Use the **[permit | deny]** <*source*> command to specify which traffic is permitted or denied.

   > **NOTE** *The **permit** and **deny** keywords do not always refer to traffic being allowed or discarded. It depends on the feature using the ACL. Instead, a match on a **permit** entry means the feature using the ACL should apply its action to this traffic, and a match on a **deny** entry means the action should not be taken.*

   Using the **no** form of this command removes the matching criteria from the ACL. You can define the <*source*> parameter using any of the following three methods:

   - Using the keyword **any** to match any IPv6 address.
   - Using **host** <*ipv6 address*> to specify a single host address. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.
   - Using <*ipv6 prefix/prefix-length*> to specify a source prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::Y/<Z**>), for example: **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**. Matches occur for sources with the same prefix bits at least to the specified length.

2. Use the **remark** <*remark*> command to associate a descriptive tag with a standard ACL. Use the **no** form of this command to remove the descriptive tag. Tags can be up to **80** alphanumeric characters enclosed in quotation marks, for example, "This list blocks all outbound web traffic."

For example, to create a standard IPv6 ACL named **OUTBOUND** that permits any traffic from a source with the same prefix bits as **2001:DB8:3F::/48**, and a remark, enter the commands as follows:

```
(config)#ipv6 access-list standard OUTBOUND
(config-std6-nacl)#permit 2001:DB8:3F::/48
(config-std6-nacl)#remark "Permit outbound traffic from LAN."
```

Creating Extended IPv6 ACLs

To create an extended IPv6 ACL, enter the **ipv6 access-list extended** <*ipv6 acl name*> command from the Global Configuration mode. This command creates and names the ACL, and enters the ACL's configuration mode. Using the **no** form of this command removes the ACL from the unit's configuration. For example, to create the extended IPv6 ACL **MATCHALL**, enter the command as follows:

```
(config)#ipv6 access-list extended MATCHALL
(config-ext6-nacl)#
```


> NOTE *If you are using both IPv4 and IPv6 ACLs, they must have different names. You cannot assign an IPv4 ACL and an IPv6 ACL the same name or you will receive an error message.*

Once you have created the IPv6 ACL and entered the ACL's configuration mode, you can specify the match criteria used by the ACL. These criteria are order-sensitive, meaning criteria are processed in the order in which they were entered. Unlike IPv4, extended IPv6 ACLs do not support log entries. The following commands specify the match criteria for the extended ACL:

1. Use the **[permit | deny]** <*protocol*> <*source*> <*destination*> **[fragments]** command to specify which traffic is permitted or denied. This command provides traffic matching based on the IPv6 header field. Using the **no** form of this command removes the matching criteria from the ACL. You can define the parameters of this command in the following methods:
   - The <*protocol*> parameter specifies which protocol this ACL entry matches. Some protocols can be entered by name (such as **ahp**, **esp**, and **gre**) or a protocol number can be entered. The keyword **ipv6** can be used to match any IPv6 traffic. Protocol number range is **0** to **255**. Extension header values are not allowed.
   - The <*source*> parameter can be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.), or using <*ipv6 prefix/prefix-length*> to specify a source IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - The <*destination*> parameter can also be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.), or using <*ipv6 prefix/prefix-length*> to specify a destination IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - The optional **fragments** parameter is used to specify that the ACL entry is only matched by non-initial fragments. The **fragments** keyword is only available when the specified protocol is **ipv6**. IPv6 ACLs match non-initial fragments in the following manner:
     - Non-initial fragments can match entries with the **fragments** keyword, provided the other Layer 3 information specified in the entry matches the packet.
     - Non-initial fragments can match entries with the **ipv6** protocol specified, provided the other Layer 3 information specified in the entry matches the packet.
     - Non-initial fragments are implicitly permitted by access groups if the fragments did not match an explicit entry in the ACL.

> NOTE *The **fragments** keyword is only useful when the ACL is referenced by an access group. The **fragments** keyword is not needed when the ACL is used only for an ACP. This is because the IPv6 firewall reassembles IPv6 fragment chains prior to inspecting the ACP to determine which action to take on the packets.*

2. Use the [**permit | deny] [tcp | udp]** <*source*> **[**<*source port*>**]** <*destination*> **[**<*destination port*>**]** **[**<*tcp flags*>**]** command to specify which traffic is permitted or denied. This command provides traffic matching based on the packet's IPv6 header fields and the upper-layer protocol flags (TCP or UDP). Using the **no** form of this command removes the matching criteria from the ACL. You can define the parameters of this command in the following methods:

   - The **[tcp | udp]** parameter specifies which upper-layer protocol headers and fields that this ACL entry matches. For example, using the **tcp** keyword specifies that TCP-specific fields and TCP flags are used in matching.
     Your options for defining source ports and destination ports will depend on whether you are using a TCP or UDP ACL.
   - The <*source*> parameter can be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.), or using <*ipv6 prefix/prefix-length*> to specify a source IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - The <*source port*> parameter optionally specifies that traffic comparison is conducted on the source port for the associated protocol (TCP or UDP). When you specify a source port, you must enter a port operator and a port number or name. Port operators include the following:
     eq <*port number/name*> matches only packets equal to a specified port number.
     gt <*port number/name*> matches only packets with a port number greater than the specified port number.
     lt <*port number/name*> matches only packets with a port number less than the specified port number.
     neq <*port number/name*> matches only packets that are not equal to the specified port number.
     range <*beginning port number/name*> <*ending port number/name*> matches only packets that contain a port number in the specified range.

     Port numbers and names for IPv6 ACLs are similar to those of IPv4 ACLs. If you are using a port range, you must enter two port values; otherwise, a single port value is used. Port number range is **0** to **65535**.
   - The <*destination*> parameter can also be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.), or using <*ipv6 prefix/prefix-length*> to specify a destination IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - The <*destination port*> parameter optionally specifies that traffic comparison is conducted on the destination port for the associated protocol (TCP or UDP). When you specify a destination port, you must enter a port operator and a port number or name. Port operators include the following:
     eq <*port number/name*> matches only packets equal to a specified port number.
     gt <*port number/name*> matches only packets with a port number greater than the specified port number.
     lt <*port number/name*> matches only packets with a port number less than the specified port number.
     neq <*port number/name*> matches only packets that are not equal to the specified port number.

range <*beginning port number/name*> <*ending port number/name*> matches only packets that contain a port number in the specified range.

Port numbers and names for IPv6 ACLs are similar to those of IPv4 ACLs. If you are using a port range, you must enter two port values; otherwise, a single port value is used. Port number range is **0** to **65535**.

- If you are using the TCP protocol, you can also optionally define which flag in the TCP header to use for traffic matching using the <*tcp flags*> parameter. These flags are defined after the destination port, and include the following choices:

  ack matches the TCP acknowledgment header flag.
  fin matches the TCP finish header flag.
  psh matches the TCP push header flag.
  rst matches the TCP reset header flag.
  syn matches the TCP synchronize header flag.
  urg matches the TCP urgent pointer header flag.

3. Use the **[permit | deny] icmpv6** <*source*> <*destination*> **[**< *message name*> **|** <*message type*> <*message code*>**]** command to specify which traffic is permitted or denied. This command provides traffic matching based on the packet's IPv6 header fields and ICMPv6-specific fields. Using the **no** form of this command removes the matching criteria from the ACL. You can define the parameters of this command in the following methods:

   - The <*source*> parameter can be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.), or using <*ipv6 prefix/prefix-length*> to specify a source IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - The <*destination*> parameter can also be specified in one of three ways: using the keyword **any** to match any IPv6 address; using **host** <*ipv6 address*> to specify a single host address (IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**), or using <*ipv6 prefix/prefix-length*> to specify a destination IPv6 prefix to match. The prefix value is specified in colon hexadecimal format (**X:X::X/<Z**>), for example, **2001:DB8:3F::/64**. The prefix length (<**Z**>) is an integer with a value between **0** and **128**.
   - You can optionally specify the ICMP message type used for matching ICMPv6 packets. ICMP messages can be defined by message name, or by entering a message type and message code. ICMPv6 message names are defined below. If you prefer to enter a message type and code, you can specify a message type by entering a value between **0** and **127** for error messages or a value between **128** to **255** for informational messages. If you enter a message type value, you must also enter a message code. Message code range is **0** to **255**. *Table 2* outlines ICMP message names.

Table 2. ICMPv6 Message Names

Message Name Entry	Message Meaning
beyond-scope	Indicates the destination is unreachable because it is beyond the scope of the source address.
dest-unreachable	Indicates the destination address is unreachable.
dhaad-reply	Indicates a home agent address discovery reply message.

Table 2. ICMPv6 Message Names *(Continued)*

Message Name Entry	Message Meaning
dhaad-request	Indicates a home agent address discovery request message.
echo-reply	Indicates an echo reply message.
echo-request	Indicates an echo request message.
header	Indicates an erroneous header field has been encountered.
hop-limit	Indicates the hop limit has been exceeded in packet transit.
mld-query	Indicates a mulitcast listener discovery query message.
mld-reduction	Indicates a multicast listener discovery reduction message.
mld-report	Indicates a multicast listener discovery report message.
mp-advertisement	Indicates a mobile prefix advertisement message.
mp-solicitation	Indicates a mobile prefix solicitation message.
nd-na	Indicates an ND Neighbor Advertisement (NA) message.
nd-ns	Indicates an ND Neighbor Solicitation (NS) message.
next-header	Indicates an unrecognized next header type was encountered.
no-admin	Indicates the destination is unreachable because communication with the destination is administratively prohibited.
no-route	Indicates the destination is unreachable because there is no route to the destination.
packet-too-big	Indicates the packet is too large.
parameter-option	Indicates that an unrecognized IPv6 option was encountered.
parameter-problem	Indicates there is a parameter problem with the packet.
port-unreachable	Indicates the destination is unreachable because the port is unreachable.
reassembly-timeout	Indicates that the fragment reassembly time limit has been exceeded.
redirect	Indicates a redirect message.
renum-command	Indicates a router renumbering command.
renum-result	Indicates a router renumbering result.
renum-seq-number	Indicates a router sequence number reset.
router-advertisement	Indicates a router advertisement message.
router-renumbering	Indicates a router renumbering for all codes.
router-solicitation	Indicates a Router Solicitation (RS) message.
time-exceeded	Indicates the time limit has been exceeded.

Table 2. ICMPv6 Message Names *(Continued)*

Message Name Entry	Message Meaning
unreachable	Indicates the destination is unreachable.

For example, to create an extended IPv6 ACL named **OUTBOUND** that permits any traffic from a source with the same prefix bits as **2001:DB8:3F::/48**, headed to a destination of **2001:DB8:85A3::8A2E:0370:7334**, and an ICMPv6 message type of **echo-request**, enter the commands as follows:

```
(config)#ipv6 access-list extended OUTBOUND
(config-ext6-nacl)#permit 2001:DB8:3F::/48 2001:DB8:85A3::8A2E:0370:7334 echo-request
```

Applying IPv6 ACLs

There are many features in AOS that use IPv6 ACLs. Once an ACL has been created, it can be used by many different IPv6 features to monitor and control feature-based traffic flow. The CLI configuration for applying ACLs allows the use of a single command to apply ACLs to either IPv4 or IPv6 traffic, using IPv6 or IPv4 ACLs and their associated keywords. For example, to apply the configured IPv6 ACL **Inboundv6** to the TFTP feature, you can use the **tftp [ip | ipv6] access-class** <*acl name*> **in** command from the Global Configuration mode prompt and specify the **ipv6** keyword to apply the ACL to IPv6 traffic only. The **ip** and **ipv6** parameters reflect recent command changes that allow you to choose either an IPv4 ACL (**ip**) or an IPv6 ACL (**ipv6**) to be used by the feature. Each keyword specifies that you are monitoring IPv4 or IPv6 traffic. For example, to apply an IPv6 ACL for inbound IPv6 TFTP traffic, enter the command as follows:

```
(config)#tftp ipv6 access-class INBOUNDv6 in
(config)#
```

To apply an IPv4 ACL for inbound IPv4 TFTP traffic, enter the command as follows:

```
(config)#tftp ip access-class INBOUNDv4 in
(config)#
```

For more information about specific commands used in applying IPv6 ACLs to specific features, refer to the *AOS Command Reference Guide*, available online at https://supportforums.adtran.com.

IPv6 Access Control Policies

ACPs are policy classes created and applied to interfaces that act on traffic entering the interface. ACPs use ACLs to determine how traffic matching the ACLs are treated within the AOS product. IPv6 ACPs function exactly the same as IPv4 ACPs, except that you must use IPv4 ACLs for IPv4 ACPs, and you must use IPv6 ACLs for IPv6 ACPs. The following commands are used in AOS to create IPv6 ACPs.

1. Use the **ipv6 policy-class** <*ipv6 acp name*> command to create an IPv6 ACP and enter the ACP's configuration mode. Using the **no** form of this command removes the ACP and all of its entries. Up to **20** IPv6 ACPs can be created. For example, to create an ACP called **PRIVATE**, enter the command from the Global Configuration mode prompt as follows:

   ```
   (config)#ipv6 policy-class PRIVATE
   (config-policy6-class)#
   ```


2. Specify the allowed traffic in the policy class using the **allow [reverse] list** <*ipv6 acl name*> **[policy** <*ipv6 acp name*> **| self] [stateless]** from the ACP's configuration mode. The **no** form of this command removes the entry from the ACP. The parameters of this command are explained below:
   - Use the optional **reverse** parameter to instruct the firewall to use the source information as the destination information and vice versa when attempting matches against the specified ACL.
   - Use the **list** <*ipv6 acl name*> parameter to specify the IPv6 ACL used by this IPv6 ACP to match traffic. The ACL specified must be an IPv6 ACL. All packets permitted by the IPv6 ACL are allowed to enter the interface to which the ACP is assigned, and an association is created in the firewall. All packets denied by the ACL are processed by the next policy class entry or are implicitly discarded if no further policy class entries exist.
   - Use the **policy** <*ipv6 acp name*> parameter to optionally specify the destination IPv6 ACP against which to match traffic. The ACP named must be an IPv6 ACP. The firewall attempts to match the specified ACP with the ACP that is applied to the packet's egress interface as determined by the routing table. If there is a match, the firewall attempts to match the ACL next. If there is no match, the firewall will process the packet based on the next policy class entry or implicitly discard it if no further policy class entries exist.
   - Use the **self** parameter to optionally specify that packets permitted by the ACL are allowed to pass when destined for any local interface on the AOS unit. These packets are terminated by the unit and are not routed or forwarded to other destinations. Using the **self** keyword is helpful when opening remote administrative access to the unit (Telnet, SSH, ICMP, Hypertext Transfer Protocol (HTTP), Secure Hypertext Transfer Protocol (HTTPS), etc.).
   - Use the **stateless** parameter to optionally specify that traffic is not subjected to built-in firewall timers. A stateless policy session will time out, but because it does not perform stateful attack checking, a new policy session for existing connections can be recreated easily.
3. Specify the discarded traffic in the policy class and create a firewall association using the **discard list** <*ipv6 acl name*> **[policy** <*ipv6 acp name*> **| self]** from the ACP's configuration mode. The **no** form of this command removes the entry from the ACP. The parameters of this command are explained below:
   - Use the **list** <*ipv6 acl name*> parameter to specify the IPv6 ACL used by this IPv6 ACP to match traffic. The ACL specified must be an IPv6 ACL. All packets permitted by the IPv6 ACL are discarded at the interface to which the ACP is assigned, and an association is not created in the firewall. All packets denied by the ACL are processed by the next policy class entry or are implicitly discarded if no further policy class entries exist.
   - Use the **policy** <*ipv6 acp name*> parameter to optionally specify the destination IPv6 ACP against which to match traffic. The ACP named must be an IPv6 ACP. The firewall attempts to match the specified ACP with the ACP that is applied to the packet's egress interface as determined by the routing table. If there is a match, the firewall attempts to match the ACL next. If there is no match, the firewall will process the packet based on the next policy class entry or implicitly discard it if no further policy class entries exist.
   - Use the **self** parameter to optionally specify that packets permitted by the ACL are discarded when destined for any local interface on the AOS unit. These packets are terminated by the unit and are not routed or forwarded to other destinations. Using the **self** keyword is helpful when opening up remote administrative access to the unit (Telnet, SSH, ICMP, HTTP, HTTPS, etc.).

For example, to create an IPv6 ACP named **UNTRUSTED** that allows any traffic matching the IPv6 ACL **INWEB** to enter the router system, enter the commands as follows:

```
(config)#ipv6 policy-class UNTRUSTED
(config-policy6-class)#allow list INWEB
```


IPv6 ACP Global Settings

In addition to creating the IPv6 ACP and specifying its match criteria, you can also set some ACP parameters globally. These parameters include setting the maximum number of allowed policy sessions in the AOS product for both IPv4 and IPv6 combined, setting the maximum number of allowed IPv6 policy sessions on a specified policy class, and verifying that traffic enters the device on the appropriate interface. The commands used for these features are described below.

1. To specify the maximum number of allowed policy sessions in the AOS product for both IPv4 and IPv6 combined, enter the **policy-class max-sessions** <*number*> command from the Global Configuration mode prompt. The <*number*> parameter is the number of allowed sessions, and the valid range is **1** to a value based on the amount of RAM in the AOS product. The default value for this command is also based on the amount of RAM in the AOS product. Default values are as follows:
   - For 64 MB of RAM, the default maximum is **10000**.
   - For 128 MB of RAM, the default maximum is **30000**.
   - For 256 MB of RAM, the default maximum is **80000**.
   - For 512 MB of RAM, the default maximum is **200000**.
   - For 768 MB of RAM, the default maximum is **300000**.
   - For 1GB of RAM, the default maximum is **450000**.

   This command specifies the limit for all policies on the AOS unit, and the **no** form of the command sets the maximum session limit to the default value. To change the default limit, enter the command as follows:

   (config)#**policy-class max-sessions 20000**
2. To specify the maximum number of allowed IPv6 policy sessions on a specific IPv6 policy class, enter the **ipv6 policy-class** <*ipv6 acp name*> **max-sessions** <*number*> command from the Global Configuration mode prompt. The <*ipv6 acp name*> parameter specifies to which IPv6 ACP the maximum session limit is applied. The specified ACP must be an IPv6 ACP. The <*number*> parameter is the number of allowed sessions, and the valid range is **1** to a value based on the amount of RAM in the AOS product. The default value for this command is also based on the amount of RAM in the AOS product. Default values are the same as in the previous command. Using the **no** form of this command sets the maximum session limit to the default value on the specified IPv6 ACP. To change the default limit for an IPv6 policy class, enter the command as follows:

   (config)#**ipv6 policy-class UNTRUSTED max-sessions 15000**
3. To specify that traffic is verified as entering the device on the proper interface, enter the **ipv6 policy-class** <*ipv6 acp name*> **rpf-check** command from the Global Configuration mode prompt. This command uses reverse path forwarding (RPF) to run a spoofing check for a specified IPv6 ACP. When enabled, after a packet is received, the firewall performs a route lookup on the packet's source address to determine what interface would be used to forward a packet back to that address. The firewall then checks the ACP assigned to that interface. If the ACP does not match the ACP of the interface on which the packet was received, the packet is dropped. Using the **no** form of this command disables the feature for the specified ACP. By default, this feature is enabled. To disable this feature on a per-IPv6 ACP basis, enter the command as follows:

   (config)#**no ipv6 policy-class UNTRUSTED rpf-check**

Applying IPv6 ACLs and ACPs to an Interface

IPv6 traffic control can be configured on a per-interface basis by applying ACLs and ACPs to the interface, just as in IPv4. The following are the IPv6 commands in AOS for interface-level traffic control.

1. Use the **ipv6 access-group** <*ipv6 acl name*> **[in | out]** command to create a static packet filter on the interface. The <*ipv6 acl name*> specifies an IPv6 ACL to use for traffic filtering, and the **in** and **out** parameters specify the direction of the traffic flow for which the filter applies. One IPv6 ACL can be applied in each direction. Unlike in IPv4, IPv6 traffic filters include an implicit **permit** for neighbor solicitation and advertisement packets in an ACL before the traditional implicit **deny** at the end of the ACL. This prevents blocking address resolution and unreachability detection, although this can be overridden by entering explicit **deny** commands in the IPv6 ACL. Using the **no** form of this command removes the traffic filter from the interface. To apply an ACL to the interface, enter the command from the interface's configuration mode as follows:

   (config)#**interface ethernet 0/1**
   (config-eth 0/1)#**ipv6 access-group PRIVATE in**

2. Use the **ipv6 access-policy** <*ipv6 acp name*> command to assign an IPv6 ACP to the interface. The assigned ACP controls sessions that are initiated in the ingress direction of the interface. Using the **no** form of this command removes the ACP from the interface. To apply an ACP to an interface, enter the command from the interface's configuration mode as follows:

   (config)#**interface ethernet 0/1**
   (config-eth 0/1)#**ipv6 access-policy UNTRUSTED**

*IPv6 ACPs will not be used to process packets arriving at the interface until the IPv6 firewall is enabled (using the **ipv6 firewall** command from the Global Configuration mode). If the IPv6 firewall is not enabled, the ACP can be configured but it will not filter traffic.*

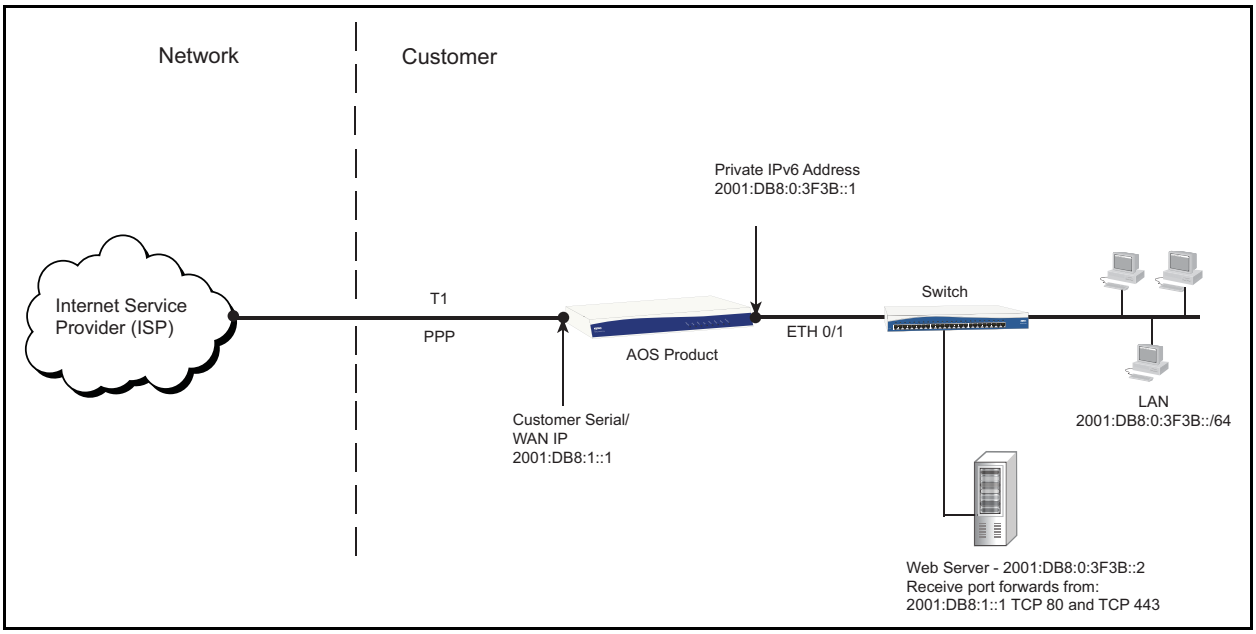
Example Configurations

Example 1 - Internet Access, Port Forward, Admin Access

In this example, an AOS router is providing Internet access for the 2001:DB8:0:3F3B::/64 subnet, as well as port forwarding HTTP and HTTPS to an internal web server (2001:DB8:0:3F3B::2). Internet access is facilitated from the **allow list** command in the **PRIVATE** ACP, which is applied to the local area network (LAN) interface of the unit (ETH 0/1). The port forward is accomplished with the **allow list** command in the **PUBLIC** ACP. Traffic matching this ACP will be directed to the internal web server at 2001:DB8:0:3F3B::2. This causes a conflict with the built-in web server on the AOS unit. To overcome this, the HTTP server is configured to run on TCP port 8080, and the HTTPS server is configured to run on TCP port 8081.

An administrative ACP has also been implemented that will only allow certain types of management traffic destined for the wide area network (WAN) interface on the unit. All traffic not matching this ACP, the port forward previously discussed, or return traffic from an open policy session on the **PRIVATE** ACP will be dropped. Allowing administrator access is accomplished with the **allow list WEB-ACL-3 self** command in the **PUBLIC** ACP. The referenced ACL (**WEB-ACL-3**) only accepts ping, SSH, TCP port 8080, and TCP port 8081 destined to the public IP of the unit (2001:DB8:1::1). The SSH lines are enabled with the **no shutdown** command. Both the HTTPS server and the SSH lines use the local user list for authentication credentials. Users will need to use the user name **admin** and the password **password** as shown in the configuration.

Users on the private side of the unit will be able to access any management services that are configured and running, as they are allowed in from the **allow list self SELF** command in the **PRIVATE** ACP. This references an ACL that allows any IP traffic.




```
username "admin" password "password"
!
ipv6 firewall
!
```



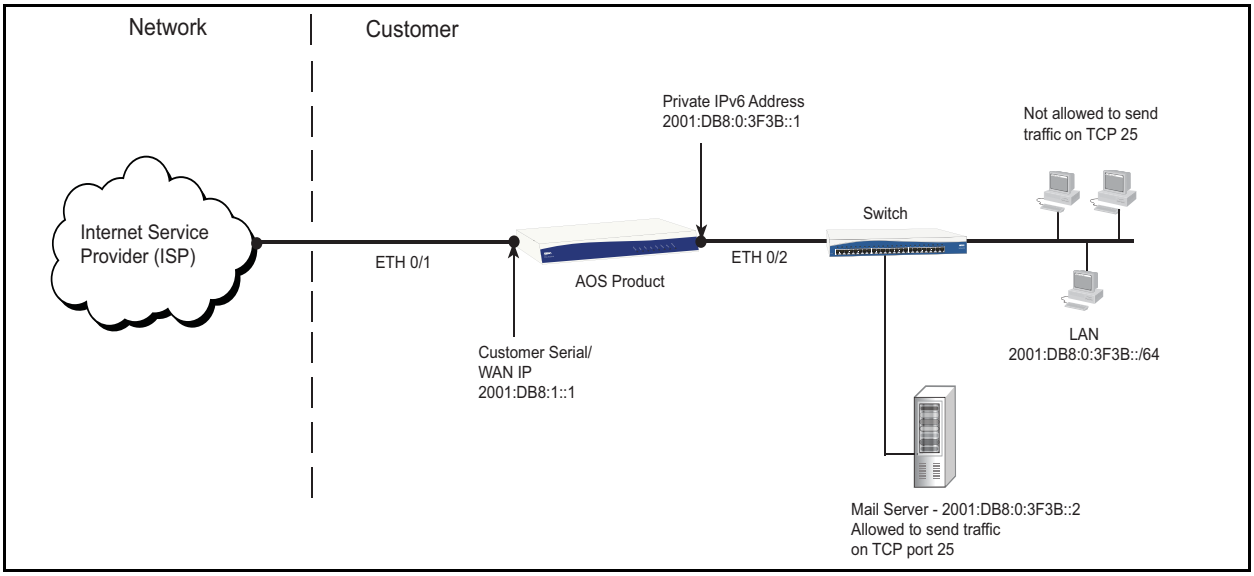
```
interface eth 0/1
  ipv6 address  2001:DB8:0:3F3B::1
  ipv6 access-policy PRIVATE
  no shutdown
!
interface t1 1/1
  clock source line
  tdm-group 1 timeslots 1-24 speed 64
  no shutdown
!
interface ppp 1
  ipv6 address  2001:DB8:1::1
  ipv6 access-policy PUBLIC
  no shutdown
  cross-connect 1 t1 1/1 1 ppp 1
!
ipv6 access-list standard WIZARD-ICS
  remark Internet Connection Sharing
  permit any
!
ipv6 access-list extended SELF
  remark Traffic to AOS product
  permit ipv6 any  any     log
!
ipv6 access-list extended WEB-ACL-3
  remark Admin Access
  permit tcp any  host 2001:DB8:1::1 eq 8080   log
  permit tcp any  host 2001:DB8:1::1 eq 8081   log
  permit tcp any  host 2001:DB8:1::1 eq ssh   log
  permit icmp any  host 2001:DB8:1::1 echo   log
!
ipv6 access-list extended WEB-ACL-4
  remark Web Server
  permit tcp any  host 2001:DB8:0:3F3B::2 eq www   log
  permit tcp any  host 2001:DB8:0:3F3B::2 eq https   log
!
ipv6 policy-class PRIVATE
  allow list SELF self
  allow list WIZARD-ICS public POLICY
!
ipv6 policy-class PUBLIC
  allow list WEB-ACL-4 policy PRIVATE
  allow list WEB-ACL-3 self
!
ipv6 route ::0/0 2001:DB8:1::2
!
http server 8080
http secure-server 8081
!
line ssh 0 4
  login local-userlist
  no shutdown
!
```

Example 2 - Filtering Ports Outbound

In a network environment, most administrators have concerns regarding outbound user traffic. Sometimes it is necessary to prevent users inside the network from accessing a specific service on the Internet. At other times, to prevent spreading viruses, only specific servers should be allowed outbound on specific ports.

This example covers a common scenario, preventing any host, other than a mail server, from sending traffic outbound on port 25. To accomplish this, an additional firewall rule must be created. The first rule in the ACP named **PRIVATE** discards any traffic that matches the ACL named **MATCH-25**. Because **MATCH-25** was defined to first deny (not process) all traffic from the mail server and then permit (process) all simple mail transfer protocol (SMTP) traffic (port 25), all SMTP traffic not from the mail server is dropped by the **discard list** command in the ACP. Users on the private side of the unit will be able to access any management services that are configured and running, as they are allowed in from the **allow list SELF self** command in the **PRIVATE** ACP. This references an ACL that allows any IP traffic.

The **PUBLIC** ACP is empty so it will deny all incoming traffic from the Internet unless it is in response to traffic initiated from hosts on one of the LANs present in the firewall session table.




```
!
ipv6 firewall
!
interface eth 0/1
  ipv6 address  2001:DB8:1::1
  ipv6 access-policy PUBLIC
  no shutdown
!
interface eth 0/2
  ipv6 address  2001:DB8:0:3F3B::1
  ipv6 access-policy PRIVATE
  no shutdown
!
ipv6 access-list extended MATCH-25
  deny ipv6 host 2001:DB8:0:3F3B::2 any
  permit tcp any any eq smtp
!
ipv6 access-list standard WIZARD-ICS
   permit any
!
ipv6 access-list extended SELF
  remark Traffic to AOS product
  permit ipv6 any any log
!
ipv6 policy-class PRIVATE
  allow list SELF self
```



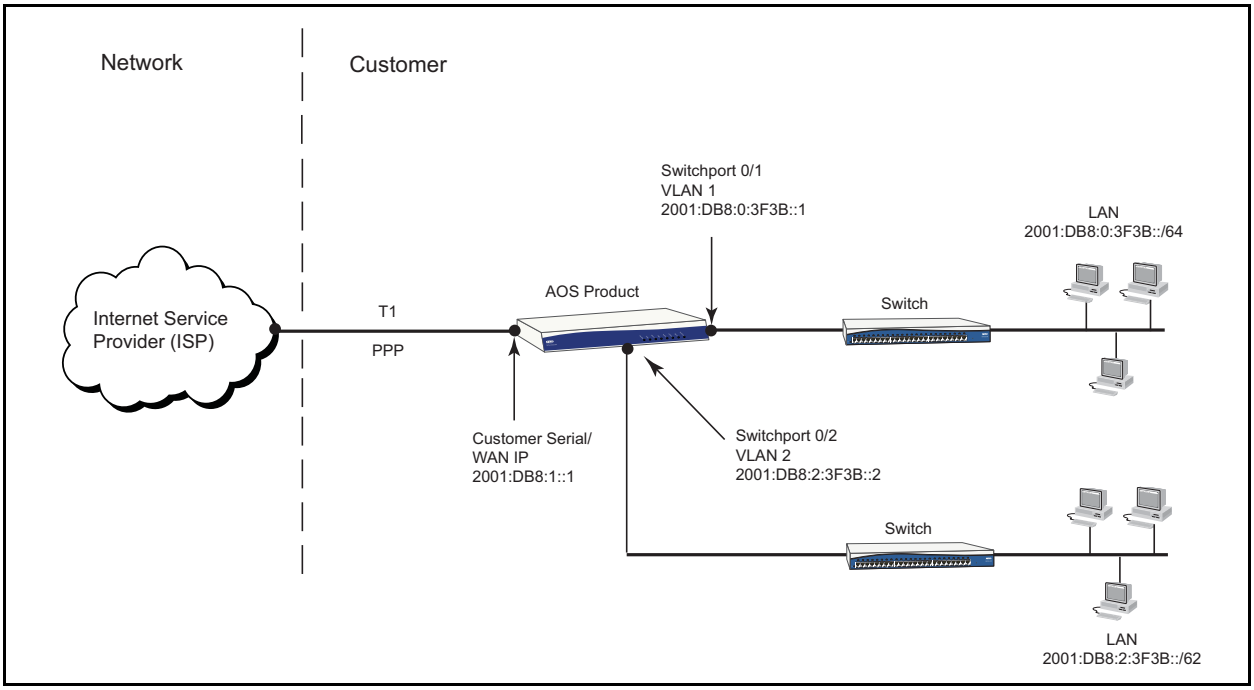
```
  discard list MATCH-25
  allow list WIZARD-ICS public POLICY
!
ipv6 policy-class PUBLIC
!Implicit discard
!
ipv6 route ::0/0 2001:DB8:1::2
!
```

Example 3 - Stateless Allow Between LAN Interfaces

In an environment where a network is segmented into multiple private subnets, either by virtual local area networks (VLANs) or multiple physical interfaces, devices in separate subnets may need to communicate with one another. To allow this communication, an **allow** ACP must be created to permit the private subnets to communicate without Network Address Translation (NAT).

In this example, an AOS unit is configured with two private VLANs, each of which is using the **PRIVATE** ACP. Since the two subnets are dissimilar and cannot be supernetted, an ACL must be created with two statements. Each ACL statement permits traffic in one direction. The ACL is then applied to the **PRIVATE** ACP as an **allow**. The **stateless** keyword is an optional parameter for the **allow list** command that prevents firewall timeouts, attack checks, and ALGs from tampering with traffic. Finally, since the ACPs are executed in sequential order, the **allow** statement must be placed first. Users on the private side of the unit will be able to access any management services that are configured and running, as they are allowed in from the **allow list SELF self** command in the **PRIVATE** ACP. This references an ACL that allows any IP traffic to self.

Only HTTPS and SSH administrative access are allowed on the **PUBLIC** ACP because it references the **ADMIN-ACCESS** ACL.




```
!
```



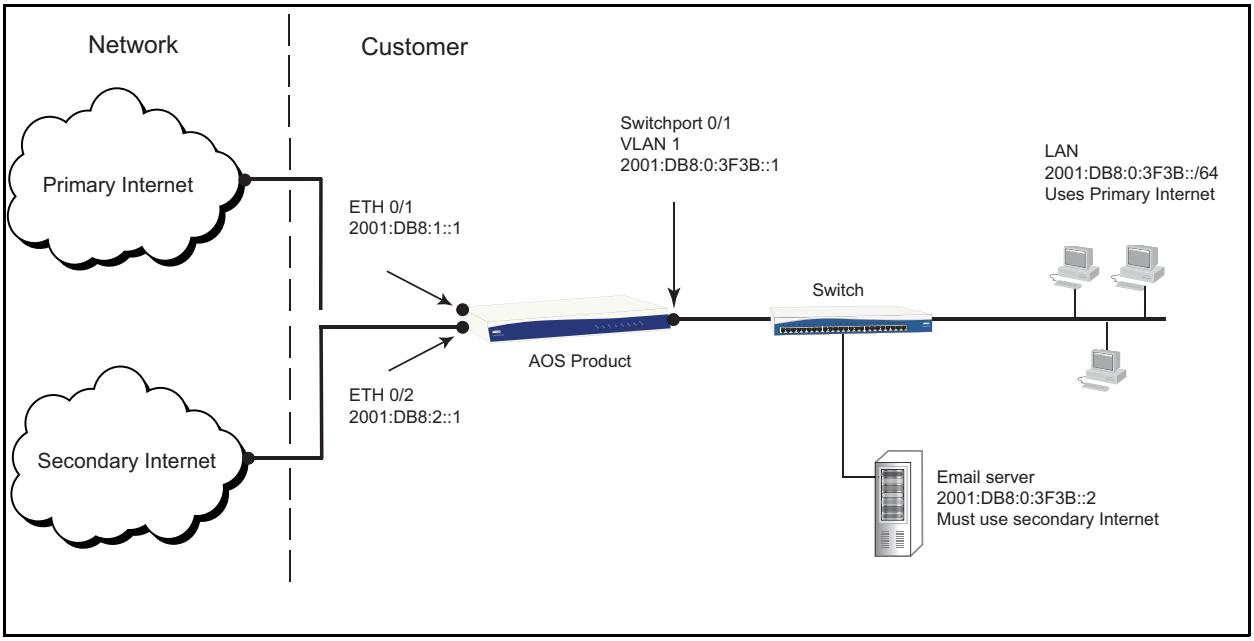
```
ipv6 firewall
!
!
interface switchport 0/1
  no shutdown
!
interface switchport 0/2
  no shutdown
  switchport access vlan 2
!
interface vlan 1
  ipv6 address  2001:DB8:0:3F3B::1
  ipv6 access-policy PRIVATE
  no shutdown
!
interface vlan 2
  ipv6 address  2001:DB8:2:3F3B::2
  ipv6 access-policy PRIVATE
  no shutdown
!
interface t1 1/1
  clock source line
  tdm-group 1 timeslots 1-24 speed 64
  no shutdown
!
interface ppp 1
  ipv6 address  2001:DB8:1::1
  ipv6 access-policy PUBLIC
  no shutdown
  cross-connect 1 t1 1/1 1 ppp 1
!
ipv6 access-list standard MATCHALL
   permit any
!
ipv6 access-list extended ADMIN-ACCESS
  permit tcp any any eq ssh
  permit tcp any any eq https
!
ipv6 access-list extended INTER-VLAN
  permit ipv6 2001:DB8:0:3F3B::/64 2001:DB8:2:3F3B::/62
  permit ipv6 2001:DB8:2:3F3B::/62 2001:DB8:0:3F3B::/64
!
ipv6 access-list extended SELF
  remark Traffic to AOS product
  permit ipv6 any any log
!
ipv6 policy-class PRIVATE
  allow list SELF self
  allow list INTER-VLAN stateless
  allow list MATCHALL policy PUBLIC
!
ipv6 policy-class PUBLIC
  allow ADMIN-ACCESS self
!
ipv6 route ::0/0 2001:DB8:1::2
!
```


Example 4 - Routing Specific Traffic to a Secondary ISP

In this example, two Internet connections are being used by an AOS unit. The primary Internet connection (ETH 0/1, 2001:DB8:1::1) is used to facilitate Internet access for all hosts on the private LAN (VLAN 1, 2001:DB8:0:3F3B::/64). The secondary Internet connection (ETH 0/2, 2001:DB8:2::1) is used strictly for email service. To prevent all the email server traffic from using the default route (which points all traffic out the primary Internet connection), the route tables must be overridden and traffic forced out the secondary Internet connection. This is accomplished with the route map **EMAIL-SERVER**, which matches all traffic coming from the server (2001:DB8:0:3F3B::2) and sends it out the secondary Internet connection. By default, the AOS firewall performs a RPF check to ensure spoofing attacks are blocked when traffic from the incorrect source IP address is received on a given interface. This check is based on active routes in the routing table that dictate what subnets exist off each interface. Because the routing table was overridden by a route map, the RPF check is no longer a valid method to check for spoofing attacks on the secondary Internet connection. The RPF check is disabled with the **no ipv6 policy-class PUBLIC-SECONDARY rpf-check** command.

The firewall checks to see what ACP is applied to the outgoing interface. This information is used to determine which firewall rule to apply to the traffic. The **PRIVATE** ACP will allow traffic to the ETH 0/1 address (2001:DB8:0:3F3B::2) only when it is going to be sent out an interface to which the **PUBLIC-PRIMARY** ACP is applied. The default route makes this true for all traffic except traffic sourced from the server, which is altered by the route map discussed above. Traffic altered by the route map will not match the first rule in the ACP because the egress interface is changed with the **set ipv6 next-hop** command to the secondary Internet connection (ETH 0/2) where the **PUBLIC-SECONDARY** ACP is applied. Due to the fact that the egress interface ACP (**PUBLIC-SECONDARY**) does not match the ACP specified in the **allow list** command (**PUBLIC-PRIMARY**), the first rule is not processed. Instead, the next rule in the **PRIVATE** ACP is evaluated. The traffic is matched on the second rule, which checks that the traffic is routed out an interface where the **PUBLIC-SECONDARY** ACP is applied. This rule will allow traffic to the ETH 0/2 address (2001:DB8:1::2) because the ACP applied to the egress interface matches the ACP specified in the second **allow list** command. Administrative access is allowed on any service from the **PRIVATE** ACP because it references the **MATCHALL** ACL to self.

Only HTTPS and SSH administrative access are allowed on the **PUBLIC-PRIMARY** and **PUBLIC-SECONDARY** ACPs because they reference the **ADMIN-ACCESS** ACL. The **PUBLIC-SECONDARY** ACP contains a rule that forwards all incoming email traffic from the secondary Internet connection to the email server (2001:DB8:0:3F3B::2).



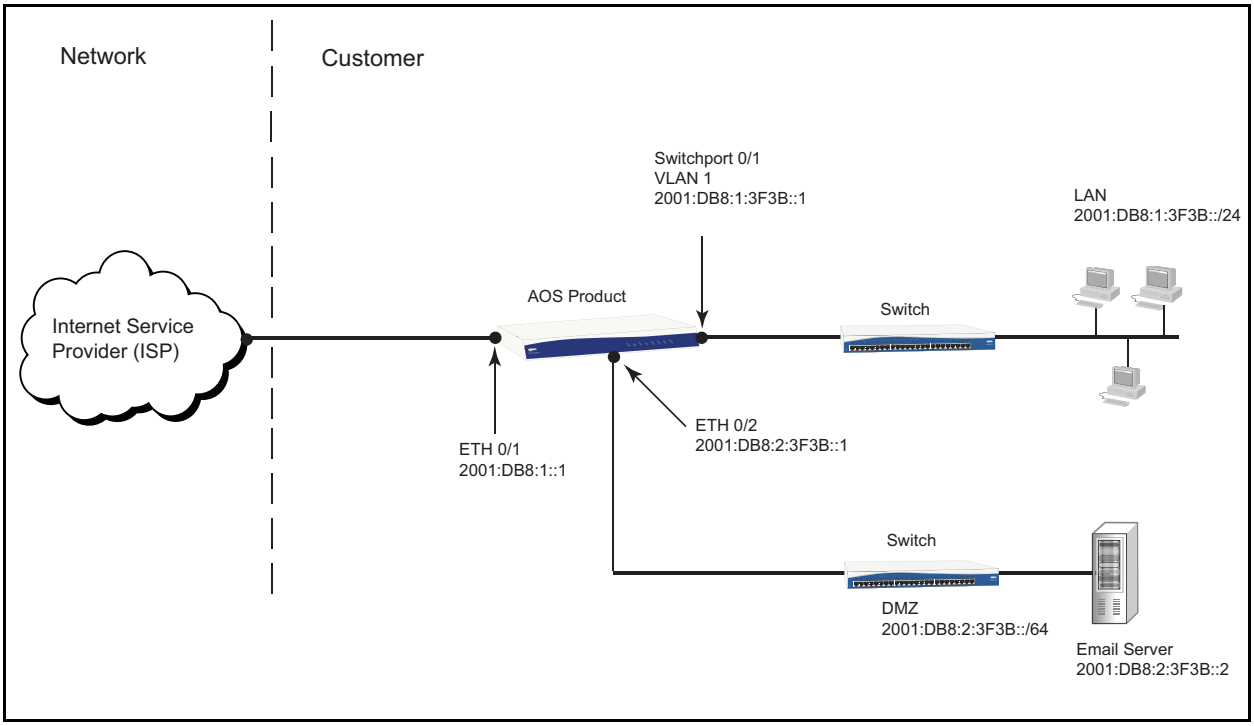

```
!
ipv6 firewall
!
interface switchport 0/1
  no shutdown
!
interface eth 0/1
  ipv6 address 2001:DB8:1::1
  ipv6 access-policy PUBLIC-PRIMARY
!
interface eth 0/2
  ipv6 address 2001:DB8:1::2
  ipv6 access-policy PUBLIC-SECONDARY
!
interface vlan 1
  ipv6 address 2001:DB8:0:3F3B::1
  ipv6 access-policy PRIVATE
  ipv6 policy route-map EMAIL-SERVER
  no shutdown
!
ipv6 route-map EMAIL-SERVER permit 10
  match ipv6 address EMAIL-OUT
  set ipv6 next-hop 2001:DB8:1::2
!
ipv6 access-list extended ADMIN-ACCESS
  permit tcp any any eq ssh
  permit tcp any any eq https
!
ipv6 access-list extended EMAIL-IN
  permit tcp any any eq smtp
!
ipv6 access-list extended EMAIL-OUT
  permit ipv6 host 2001:DB8:0:3F3B::2 any
!
ipv6 access-list extended MATCHALL
  permit ipv6 any any
!
```



```
ipv6 policy-class PRIVATE
  allow list MATCHALL self
  allow list MATCHALL interface eth 0/1 overload policy PUBLIC-PRIMARY
  allow list MATCHALL interface eth 0/2 overload policy PUBLIC-SECONDARY
!
ipv6 policy-class PUBLIC-PRIMARY
  allow list ADMIN-ACCESS self
!
no ipv6 policy-class PUBLIC-SECONDARY rpf-check
ipv6 policy-class PUBLIC-SECONDARY
  allow list ADMIN-ACCESS self
  allow list EMAIL-IN address 2001:DB8:0:3F3B::2
!
ipv6 route ::0/0 2001:DB8:1::2
!
```

Example 5 - Configuring a Privately Addressed DMZ

In this example, a demilitarized zone (DMZ) is created using private addresses to segregate an email server from the rest of the LAN. Public access servers are intentionally made available to traffic on the Internet to offer a service (for example, Internet access, email, FTP, etc.). Servers running application services are more susceptible to application-specific exploits and should be unable to initiate traffic to the LAN. Hosts in the DMZ (2001:Db8:0:3F3B::/64) only have the ability to initiate traffic to other hosts in the DMZ or out to the Internet. However, hosts on the LAN (2001:DB8:0:3F3B::/24) are able to initiate traffic to hosts in the DMZ to use the services they offer. The AOS firewall allows traffic from the DMZ back to the LAN only if it is in response to traffic sourced from the LAN. For instance, anyone on the LAN is able to access the mail server, but if the mail server is compromised by a hacker, it will not have access to anything on the LAN because the **DMZ** ACP drops the traffic to prevent it from accessing internal resources. This is accomplished with the **discard list** statement in the **DMZ** ACP. It references the **MATCHALL** ACL with the ACP **PRIVATE** keyword. This will match all traffic initiated from the DMZ and destined for the LAN where the **PRIVATE** ACP is applied. The resulting traffic will be dropped because it is a **discard list** statement. The **allow list** command in the **DMZ** ACP facilitates Internet access for any outbound traffic from the DMZ. Inbound mail traffic is forwarded to the email server using the **allow list** command in the **PUBLIC** ACP. Administrative access is restricted to HTTPS and SSH on the **PUBLIC** ACP, but is open for any type of administrative traffic on the **PRIVATE** ACP.




```
!
ipv6 firewall
!
interface switchport 0/1
  no shutdown
!
interface vlan 1
  ipv6 address  2001:DB8:1:3F3B::1
  ipv6 access-policy PRIVATE
  no shutdown
!
interface eth 0/1
  ipv6 address  2001:DB8:1::1
  ipv6 access-policy PUBLIC
  no shutdown
!
interface eth 0/2
  ipv6 address  2001:DB8:2:3F3B::1
  ipv6 access-policy DMZ
  no shutdown
!
ipv6 access-list standard ANY
   permit any
!
ipv6 access-list extended ADMIN-ACCESS
  permit tcp any any eq ssh
  permit tcp any any eq https
!
ipv6 access-list extended MAIL
  permit tcp any  any eq smtp
!
ipv6 access-list extended SELF
  permit ipv6 any  any     log
!
```



```
ipv6 policy-class PRIVATE
  allow list SELF self
  allow reverse list MATCHALL policy DMZ stateless
  allow list MATCHALL policy PUBLIC
!
ipv6 policy-class PUBLIC
  allow list MAIL policy DMZ
  allow list ADMIN-ACCESS self
!
ipv6 policy-class DMZ
  discard list MATCHALL policy PRIVATE
  allow list MATCHALL policy PUBLIC
!
ipv6 route ::0/0 2001:DB8:1::2
!
```

Command Summary Table

The following table summarizes the minimum steps required to configure the IPv6 firewall on an AOS product.

Table 3. Firewall Configuration Steps

Step	Command	Description
Step 1	Boot up the unit, then telnet to the unit (**telnet** *<ipv6 address>*)	Access the CLI.
Step 2	(config)#**ipv6 firewall [local-traffic-only]**	Enable the IPv6 firewall functionality in AOS. Optionally enables local traffic processing only (**local-traffic-only**).
Step 3	(config)#**ipv6 access-list [extended | standard]** *<ipv6 acl name>*	Create a standard or extended ACL and enter the IPv6 ACL configuration mode.
	(config-std6-nacl)#**[permit | deny]** *<source>*	If you created a standard ACL, specify the packet source IP information and decide whether the ACL will permit or deny traffic based on a match.
	(config-ext6-nacl)#**[permit | deny]** *<protocol> <source> <source port> <destination> <destination port>*	If you created an extended ACL, specify whether the ACL will **permit** or **deny** traffic based on protocol, source IP information, or destination IP information.
(Optional)	(config-std6-nacl)#**remark** *<remark> or* (config-ext6-nacl)#**remark** *<remark>*	Associate a comment with the entry to further describe the purpose of the standard or extended ACL.

Table 3. Firewall Configuration Steps *(Continued)*

Step	Command	Description
Step 4 Allow Traffic Based on ACL Entries Discard Traffic Based on ACL Entries	(config)#**ipv6 policy-class** <*ipv6 acp name*>	Create an IPv6 ACP and enter the ACP configuration mode.
	(config-policy-class)#**allow list** <*ipv4 acl name*> **[self | policy** <*ipv4 acp name*>**] [stateless]**	Specify an ACL to determine which IP packets are allowed to enter the interface to which the IPv6 ACP is assigned, and create a policy session in the firewall.
	(config-policy6-class)#**allow reverse list** <*ipv6 acl name*> **[self | policy** <*ipv6 acp name*>**] [stateless]**	Specify an ACL to determine which IP packets are allowed to enter the interface to which the IPv6 ACP is assigned, and create a policy session in the firewall. The **reverse** keyword instructs the firewall to use the source information as the destination information and vice versa when attempting matches against the specified ACL.
	(config-policy6-class)#**discard list** <*ipv6 acl name*> **[self | policy** <*ipv6 acp name*>**]**	Specify an ACL to determine which packets are discarded after being received on the interface to which the IPv6 ACP is assigned.
Step 5	(config)#**interface** <*interface*> (config-interface)#**ipv6 access-policy** <*ipv6 acp name*>	Enter the appropriate interface configuration mode, and apply the IPv6 ACP to the interface.

Troubleshooting

There are a number of methods available to assist you in troubleshooting the IPv6 firewall configuration on your unit. The following list provides methods available through the CLI.

- The **show** commands provide a means to verify your configuration (refer to *Using Show Commands* below).
- The **clear** commands clear statistics for the associated feature or clear traffic flows (refer to *Using Clear Commands on page 49*).
- The **debug** commands display events as they occur to help you better interpret possible problems (refer to *Enabling Debug Commands on page 50*).

Using Show Commands

After configuring the IPv6 firewall, several **show** commands can be issued in the CLI to assist in troubleshooting and verifying your configuration. Enter **show** commands from the Enable prompt or use the **do show** command from any configuration prompt. The following section lists these commands and their descriptions.

Use the **show ipv6 access-list [**<*ipv6 acl name*>**]** command to display the statistics for all configured IPv6 ACLs or for a specified IPv6 ACL. Enter the command as follows:

```
#show ipv6 access-list PRIVATEv6
Extended IPv6 access-list PRIVATEv6
    deny tcp any eq telnet any (0 matches)
    deny tcp any any eq telnet (0 matches)
```



```
permit ipv6 any host 2000:1::1 (0 matches)
permit ipv6 host 2000:2::1 any (0 matches)
permit icmpv6 any any (0 matches)
```

Use the **show run ipv6 access-list** command to display the IPv6 ACLs in the unit's running configuration. Enter the command as follows:

```
#show run ipv6 access-list
    ipv6 access-list extended PRIVATEv6
    deny tcp any eq telnet any
    deny tcp any any eq telnet
    permit ipv6 any host 2000:1::1
    permit icmp any any
```

Use the **show ipv6 policy-sessions [pending] [**<*ipv6 acp name*> **| any-vrf | vrf** <*name*>**]** command to display a list of current policy sessions in the firewall. Current associations are active sessions allowed through the firewall. The optional **pending** parameter specifies that pending sessions (rather than active ones) are displayed. The optional <*ipv6 acp name*> parameter limits the output to policy sessions created using the specified ACP. The optional **vrf** <*name*> parameter specifies the particular firewall instance for which active policy sessions will be displayed. If no **vrf** <*name*> is specified, policy session are displayed for the default (unnamed) VRF only. The optional **any-vrf** parameter specifies that all policy sessions for all instances of the firewall for all VRFs are displayed. If no options are defined, all current ACP associations are displayed (including locally processed traffic). Enter the command as follows:

```
#show ipv6 policy-sessions
NOTE: The "Layer 4" info below for TCP and UDP is source port and dest port. For ICMPv6, it is ID
and type/code. For all other protocols, it is unused.

Src VRF (if not default), Src policy-class:
Protocol (TTL) -> Dest VRF, Dest policy-class
    Src IPv6 Address                Layer 4
    Dest IPv6 Address               Layer 4
    ------------------------        ----------

IPv6 policy-class PRIVATEV6:
icmpv6 (59) -> self
    2001:DB8:1:1::2                 0
    2001:DB8:1:1::1                 128/0
```

Use the **show run ipv6 policy-class** command to display IPv6 ACPs in the unit's running configuration. Enter the command as follows:

```
#show run ipv6 policy-class
    ipv6 policy-class UNTRUSTED
    allow list LOCALSERVICEv6
    discard list WEBTRAFFIC
```


Using Clear Commands

The **clear** commands are issued from the Enable mode prompt and clear all statistics associated with the specified feature. The **clear** commands associated with the IPv6 firewall are described in the following section.

- Use the **clear ipv6 access-list [**<*ipv6 acl name*>**]** command to clear the statistics for all the configured IPv6 ACLs or for a specified ACL. To clear the statistics for all IPv6 ACLs, enter the command as follows:

  #**clear ipv6 access-list**

Use the **clear ipv6 policy-sessions [pending] [any-vrf | vrf** <*name*>**]** command to clear sessions from the firewall association database. Clearing a session typically terminates the session's communication, therefore this command should be used carefully, particularly if the session is the one used for access to the CLI. Using the optional **pending** parameter specifies that pending sessions (rather than active ones) are cleared. Using the optional **any-vrf** parameter specifies that all sessions in all VRFs are cleared. Using the optional **vrf** <*name*> parameter specifies that the sessions of the specific VRF are cleared. If no VRF is specified, sessions from the default (unnamed) VRF are cleared.

This command can also be used to clear a specific policy session. The exact session must be identified using the full session specification. Use one of the following commands:

- **clear ipv6 policy-sessions [vrf** <*name*>**]** <*ipv6 acp name*> **[ahp | esp | gre |** <*protocol*>**]** <*source ipv6 address*> <*destination ipv6 address*>
- **clear ipv6 policy-session [vrf** <*name*>**]** <*ipv6 acp name*> **[tcp | udp]** <*source ipv6 address*> <*source port*> <*destination ipv6 address*> <*destination port*>
- **clear ipv6 policy-sessions [vrf** <*name*>**]** <*ipv6 acp name*> **icmpv6** <*source ipv6 address*> <*id*> <*destination ipv6 address*> <*type/code*>

The **ahp**, **esp**, **gre**, **tcp**, **udp**, and **icmpv6** parameters specify a specific type of protocol session to clear. The <*protocol*> parameter allows you to specify a protocol by number. Valid protocol range is **0** to **255**. The <*source ipv6 address*> and <*destination ipv6 address*> specify the source and destination IPv6 addresses for the session you are clearing. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**.

*Whenever a link-local IPv6 address is entered (an address beginning with **FE80::**), an interface name must be entered after it.*

The <*source port*> and <*destination port*> parameters specify the source and destination ports for the TCP or UDP session. Port ranges are **0** to **65535**. The <*id*> parameter is the ICMPv6 ID. Valid ID range is **0** to **65535**. The <*type/code*> parameter specifies the type and code for the ICMPv6 session. Type and code ranges are **0** to **255**.

To clear IPv6 policy sessions on the VRF **RED**, enter the command as follows:

#**clear ipv6 policy-sessions vrf RED**

Use the **clear ipv6 policy-stats [**<*ipv6 acp name*> | <*ipv6 acp name*> **entry** <*number*>**]** command to clear statistics for IPv6 ACPs. Statistics for all IPv6 ACPs can be cleared, all entries within an ACP can be cleared, or only specific entries within an ACP can be cleared. The **entry** <*number*> command allows you to clear a specific entry. Number range is **1** to **4294967295**. The <*ipv6 acp name*> parameter allows you to specify an IPv6 ACP. For example, to clear the statistics for the sixth entry in the ACP **PRIVATE**, enter the command as follows:

#**clear ipv6 policy-stats PRIVATE entry 6**

Enabling Debug Commands

Another method for troubleshooting the IPv6 firewall configuration is enabling and viewing **debug** commands. These commands are issued from the Enable mode prompt, and are displayed (real time) to the terminal (or Telnet) screen. The **debug** commands associated with the IPv6 firewall are described in the following section.

Turning on a large amount of debug information can adversely affect the performance of your unit.

Use the **debug ipv6 firewall [ndar] [vrf** <*name*>**]** command to enable debug messages for IPv6 firewall processing. You can use the optional **ndar** parameter to only display information about ND address resolution. You can also optionally limit the output of this command to events on a single VRF using the **vrf** <*name*> parameter. If no VRF is specified, IPv6 firewall processing messages for the default (unnamed) VRF are displayed. Enter the command as follows:

```
#debug ipv6 firewall vrf Gray
2010.08.17 20:39:25 FIREWALL_V6.VRF gray id=firewall time="2010-08-17 20:39:25"
fw=NV3430 pri=6 proto=icmpv6 src=2001:DB8:1:1::2 dst=2001:DB8:1:1::1 msg="ICMPv6
type=128 code=0; Bytes processed over policy-session (bytes=256) from PRIVATEV6
policy-class on the interface eth 0/1.1"
2010.08.17 20:39:25 FIREWALL_V6.VRF gray Deleted policy-session due to clear all policy-sessions
command
Policy-session ID = 2; Count (total, policy-class) = 0, 0
Protocol = 58; Flags = 0x1; Timeout = 60
Initiating side: Policy-class = PRIVATEV6
    From/To: eth 0/1.1 -> Loopback
    Source: 2001.DB8:1:1::2
    Destination: 2001:DB8:1:1::1
    ICMPv6 Type/Code: 128/0; ID: 0
Responding side: Policy-class = self
    From/To: Loopback -> eth 0/1.1
    Source: 2001:DB8:1:1::2
    Destination: 2001:DB8:1:1::2
    ICMPv6 Type/Code: 129/0; ID: 0
```


Additional References

The information listed in *Table 4* is additional IPv6 documentation provided by ADTRAN, available online at https://supportforums.adtran.com.

Table 4. Additional AOS IPv6 Documentation

Subject	Document Title/Article Number
IPv6 Border Gateway Protocol	*Configuring BGP in AOS for Releases 18.03.00/R10.1.0 or Later*
DHCPv6	*Configuring DHCPv6 in AOS*
IPv6 in AOS	*Configuring IPv6 in AOS*
IPv6 Quality of Service	*Configuring QoS in AOS*
IPv6 VRRPv3	*IPv6 VRRPv3 for AOS*
OSPFv3	*OSPFv3 in AOS*

The information listed in *Table 5* includes the packet checks automatically performed by the IPv6 firewall that cannot be enabled, disabled, or otherwise configured. These checks do result in printed attack log messages.

Table 5. Firewall Packet Checks Resulting in Attack Log Messages

No route to destination (includes non-IPsec packets, post-IPsec packets, ICMP errors, ICMP responses)
Found non-OSPF upper layer PDU after using inbound OSPFv3 IPsec SA
Invalid IP header version
Payload length=0, NH!=No-Next-Header
Packet shorter than specified extension header length
Excessive options padding (Pad1)
Multiple pad option back-to-back (e.g., PadN followed by Pad1 or vise-versa)
Excessive options padding (PadN)
Non-zero pad
Destination option encountered in hop-by-hop options header
Unsupported option encountered
Hop-by-hop option encountered in destination options header
Duplicate router alert options in hop-by-hop header
Jumbo payload option found in packet with non-zero payload length
Invalid option length
Invalid option alignment
Packet too short for Fragment header

Table 5. Firewall Packet Checks Resulting in Attack Log Messages *(Continued)*

Packet too short for ESP header
Authentication header short
Authentication header length too short or packet shorter than specified length
Authentication header length not multiple of eight octet units
Excessive TCP options padding
Non-zero TCP options padding
Truncated TCP option (no length field)
Truncated TCP option
MSS option found but SYN flag not set
Invalid length for TCP option
ICMP error with invalid destination
ICMP error does not contain entire offending packet (up to a minimum IPv6 MTU) or offending packet IP header length is too short
ICMP error destination does not match offending packet source
Parameter problem ICMP error should not be sent for this option type
ICMP error in response to packet with multicast destination
ICMP error in response to Land attack
ICMP echo request or reply with non-zero code
Neighbor discovery with non-zero code
Invalid hop limit for neighbor discovery
Multicast listener discovery with non-zero code
Invalid hop limit for multicast listener discovery
Packet too short for GRE header
Packet too short for GRE header specified by flags
GRE version 1 (PPTP) with non-PPP protocol
ICMP error contains truncated extension header in offending packet (no length field)
ICMP error contains truncated extension header in offending packet
ICMP error contains truncated TCP header in offending packet
ICMP error contains truncated UDP header in offending packet
ICMP error contains truncated ICMP header in offending packet
ICMP error in response to ICMP error or redirect
ICMP error in response to unsupported ICMP type
ICMP error contains truncated ESP header in offending packet
ICMP error contains truncated AH header in offending packet

Table 5. Firewall Packet Checks Resulting in Attack Log Messages *(Continued)*

Source and/or destination address is IPv4-mapped
Source and/or destination address is IPv4-compatible
Packet with source of unspecified address cannot be forwarded
Loopback address used for non-local traffic
Link local addressed packet must be coming from or destined to SELF
Neighbor Discovery must be sourced from or destined to SELF
ICMP error for Neighbor Discovery that is not sourced form or destined to SELF

