

Configuration Guide

Configuring IPv6 in AOS

This configuration guide describes Internet Protocol version 6 (IPv6) and its use in ADTRAN Operating System (AOS) products. This guide provides a basic overview of IPv6 functionality and an overview of how IPv6 functions in AOS, as well as includes descriptions of command line interface (CLI) commands used by AOS for specific IPv6 feature configurations.

This guide contains the following sections:

- [*IPv6 Overview on page 2*](#)
- [*IP Addresses in IPv6 on page 5*](#)
- [*IPv6 Route Cache and Traffic Processing on page 8*](#)
- [*Hardware and Software Requirements and Limitations on page 10*](#)
- [*IPv6 Implementation in AOS on page 12*](#)
- [*IPv6 and the Interface in AOS on page 13*](#)
- [*Configuring IPv6 Routing in AOS on page 16*](#)
- [*IPv6 IPsec on page 19*](#)
- [*IPv6 General Utility Commands on page 33*](#)
- [*IPv6 Neighbor Discovery \(ND\) in AOS on page 39*](#)
- [*IPv6-Aware HTTP Server in AOS on page 48*](#)
- [*IPv6 DNS in AOS on page 51*](#)
- [*ICMPv6 in AOS on page 54*](#)
- [*NTP over IPv6 on page 55*](#)
- [*IPv6 Traffic Control in AOS on page 57*](#)
- [*Configuring IPv6 Traffic to Traverse an IPv4 GRE Tunnel on page 69*](#)
- [*Additional Resources on page 79*](#)

IPv6 Overview

IPv6 is the next generation of Internet protocols designed to replace IPv4. Both IPv4 and IPv6 are packet protocols designed to pass data, voice, and video over digital networks. IPv4 is the current standard used for Internet communication, however, its available address space is quickly diminishing. As new global markets gain access to the Internet and as new devices, particularly mobile devices, are connected to the Internet, IPv4 32-bit addressing limits the number of available IP addresses for these new markets and new devices. To combat the IP addresses crisis, IPv6 provides IP addresses based on 128 bits (rather than the 32-bit IPv4), providing enough IP addresses for the foreseeable future. IPv6 not only provides many more addresses than IPv4, it also restores end-to-end communication because the use of network address translation (NAT) is unnecessary, provides more efficient packet forwarding, and contains within itself support for Internet security and mobility. IPv6 also provides improved option support and mandated security for Internet traffic when compared with IPv4. The push for IPv6 deployment has taken a much stronger turn in the past few years, and although the switch from IPv4 to IPv6 will be a lengthy process, IPv6 is quickly becoming an affordable option for replacing IPv4 in many networks.

Although IPv6 is similar to IPv4, changes in the basic packet structure have been made between IPv4 and IPv6. IPv6 packets have a simpler header construction than IPv4, which provides more routing efficiency, performance, and forwarding rate scalability. IPv6 also functions in a slightly different manner than IPv4, replacing Address Resolution Protocol (ARP) with neighbor discovery (ND) protocols in Internet Control Management Protocol version 6 (ICMPv6), reducing the need for NAT, and using different addressing schemes, terminology, and routing processes. The following sections provide brief overviews of the differences between IPv4 and IPv6.

IPv4 and IPv6 Comparison

There are a few differences in the overall operation of IPv4 and IPv6. [Table 1](#) outlines the major differences between the two protocols. These differences are outlined in detail throughout this guide.

Table 1. Overview Comparison of IPv4 and IPv6

Feature	IPv4	IPv6
IP Address Length	32 bits	128 bits
IP Security (IPsec) Header Support	Optional	On through traffic
Prioritized Delivery Support	Some	Expanded
Packet Fragmentation	Performed by hosts and routers	Performed by hosts only
Minimum MTU	576 bytes	1280 bytes
Checksum in Packet Header	Yes	No
Options in Packet Header	Yes	No
Link-Layer Address Resolution	ARP (broadcast)	Multicast ND messages
Multicast Membership Protocol	Internet Group Management Protocol (IGMP)	Multicast Listener Discovery (MLD)

Table 1. Overview Comparison of IPv4 and IPv6 (Continued)

Feature	IPv4	IPv6
Router Discovery	Optional	Required
Uses Broadcast Messages	Yes	No
Configuration	Manual, Dynamic Host Control Protocol (DHCP)	Manual, Automatic, DHCP version 6 (DHCPv6)
Domain Naming System (DNS) Queries	Uses A records	Uses AAAA records
DNS Reverse Queries	Uses IN-ADDR.ARPA	Uses IP6.ARPA

IPv6 Terminology

IPv6 has many concepts that differ from IPv4. [Table 2](#) outlines some of the useful terminology used in IPv6 processes.

Table 2. IPv6 Terminology

Term	Definition
Node	A device that implements IP.
Router	A node that forwards IP packets not explicitly addressed to itself.
Host	Any node that is not a router.
Link	A communication facility or medium over which nodes can communicate at the link layer (Layer 2) directly below the IP layer (Layer 3).
Interface	A node's attachment to a link.
Neighbors	Nodes attached on the same link.
On-Link IP Address	An IP address that is assigned to an interface on a specified link. These addresses can be accessed directly on the link. Nodes consider IP addresses to be on-link if it is covered by one of the link's prefixes (indicated by the on-link flag in the prefix area of the packet header) or if a neighboring router specifies the IP address as the target of a redirect message.
Off-Link IP Address	An IP address that is not assigned to any interfaces on the specified link. These addresses must be accessed using a router.

IPv6 Headers

IPv6 headers differ from IPv4 headers. In IPv6, the Internet header length (IHL), Identification, Flags, Fragment Offset, Header Checksum, Options, and Padding fields are not included in the packet header. Just like IPv4, IPv6 does include the Version, Source Address, and Destination Address fields. Some fields, like the IPv4 header fields Type of Service (ToS), Total Length, Time to Live (TTL), and Protocol headers have different names and positions in IPv6. In addition, IPv6 packet headers include the following new fields of Traffic Class, Flow Label, Payload Length, Next Header, and Hop Limit. The IPv6 packet header is shown in *Figure 1*, and *Table 3* outlines the function of each header field in the IPv6 packet.

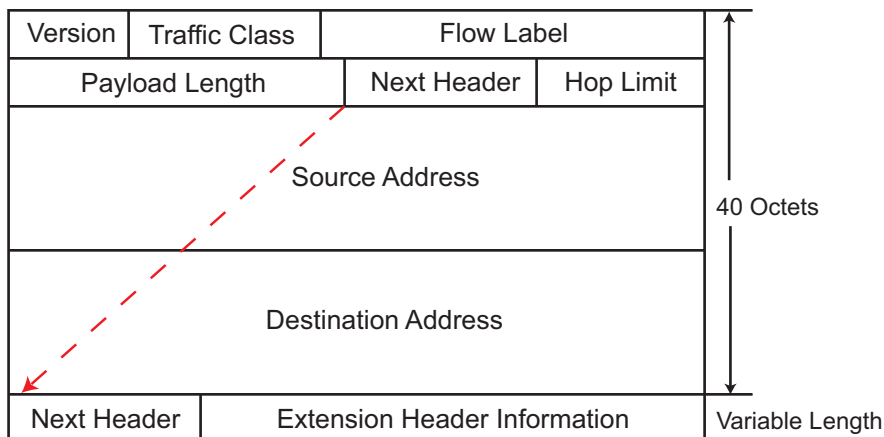


Figure 1. IPv6 Packet Header Fields

Table 3. IPv6 Packet Header Fields and Their Function

Field Name	Function
Version	Specifies the IP version. This field is the same as in IPv4 packet headers.
Traffic Class	Defines the relative priority of the packet. This field replaces the ToS header in the IPv4 packet, however, it uses the same definitions as the IPv4 header (for example, DIFFSERV code points).
Flow Label	Identifies all packets that belong to a single flow, allowing routers to treat the packets in a similar fashion.
Payload Length	Specifies the length of the payload in the IPv6 packet. This field replaces the Total Length field in an IPv4 packet.
Next Header	Indicates the next Extension header to be examined by the device. Extension headers are used in IPv6 to encode additional IP layer information. This field replaces the Protocol header in an IPv4 packet.
Hop Limit	Indicates the maximum number of hops allowed. This field replaces the TTL header in an IPv4 packet.
Source Address	Identifies the packet’s source node IP address. This header field is carried over from IPv4.
Destination Address	Identifies the packet’s destination node IP address. This header field is carried over from IPv4.

Extension Headers in IPv6

An additional difference in IPv4 and IPv6 headers is that the IPv6 packets include extension headers as a method of presenting additional packet and protocol information and options. This additional information is contained between the IPv6 header and the upper-layer header in a packet and allows you to chain extension headers together using the Next Header field. The Next Header field in IPv6 indicates to the router which extension header should be expected next. If there are no additional extension headers, the Next Header field in the IPv6 packet indicates the upper-layer header instead. These upper-layer headers include Transmission Control Protocol (TCP) headers, User Datagram Protocol (UDP) headers, ICMPv6 headers, encapsulated IP packets, etc.

Extension header options are examined at the destination only, except for hop-by-hop options. Extension headers include the following options (in this order):

1. Hop-by-Hop Options Header: Reviewed by all nodes in the packet path.
2. Destination Options Header: Reviewed by intermediate destinations along the packet path when the Routing Header is present.
3. Routing Header: Used to determine the packet path along intermediate nodes before the final packet destination.
4. Fragment Header: Used to break the packet into fragments and put the fragments back together at the final destination. These fragments allow packets larger than the path maximum transmission unit (MTU) to pass.
5. Authentication Header: Part of IPsec. This header functions exactly as in IPv4.
6. Encapsulating Security Payload Header: Part of IPsec. This header functions exactly as in IPv4.
7. Destination Options Header: Reviewed by the final packet destination.

IP Addresses in IPv6

The largest difference between IPv4 and IPv6 is the IP addressing scheme. In IPv4, IP addresses are based on 32 bits and are expressed in the dotted decimal notation format (**xx.xx.xx.xx**), for example, **10.10.10.1**. IPv6, on the other hand, uses 128 bits for IP addresses and expresses these addresses in colon hexadecimal notation (**XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX:XXXX**), for example, **2001:DB8:1::1**.

IPv6 addresses are assigned to interfaces, and they use a concept called the Interface ID which is used to uniquely identify interfaces on a link. Interface IDs make up the host portion of the IPv6 address, or the last 64 bits of the address, for unicast and anycast IPv6 addresses. Using an Interface ID helps to automate IPv6 addressing, and each Interface ID can be generated in a number of ways, including using a 64-bit extended unique identifier (EUI-64), basing the Interface ID on the link-layer address of the interface, or using a randomly generated value. The EUI-64 format for IPv6 addresses and Interface IDs is constructed from IEEE 802 addresses (this is the medium access control (MAC) address) and is applied to IEEE 802 interface types (for example, an Ethernet interface). When the EUI-64 format is used, the first 24 bits of the address are taken from the MAC address of the interface, the next 16 bits are given the hexadecimal value of FFFE, and the last 24 bits are taken from the last three octets of the MAC address. Lastly, EUI-64 based Interface IDs set the Universal/Local bit (the seventh bit of the first octet) to either **0** (for a locally administered identifier) or to **1** (for a globally unique identifier). The following example outlines the generation of an EUI-64 Interface ID from a MAC address:

If the MAC address of the interface is 00-AA-00-3F-2A-1C, follow these steps to create an EUI-64 IPv6 address (link-local address):

1. Covert the MAC address to the EUI-64 format. In this example, 00-AA-00-3F-2A-1C becomes 00-AA-00-FF-FE-3F-2A-1C.
2. Complement the U/L bit, making the address 02-AA-00-FF-FE-3F-2A-1C.
3. Convert the new address to colon hexadecimal format. In this example, the address becomes ::2AA:FF:FE3F:2A1C.
4. The link-local address for this node with the MAC address of 00-AA-00-3F-2A-1C becomes FE80::2AA:FF:FE3F:2A1C.

Other interface types can be given IPv6 addresses based on a randomly generated value (a temporary IPv6 address), a value assigned by a stateful address configuration protocol, such as DHCPv6 (where the server picks the value), or a manually configured value.

There are a few conventions used with expressing IPv6 addresses. In IPv6 addresses, leading zeros can be compressed and two colons can be used when consecutive 16-bit values are zero within the address. This helps IPv6 addresses to be less cumbersome. [For example](#), the address 2001:0DB8:0000:2F3B:02AA:00FF:FE28:9C5A becomes 2001:DB8:0:2F3B:2AA:FF:FE28:9C5A when the leading zeros are compressed. When double colons are used, an IPv6 address such as FE80:0:0:0:2AA:FF:FE9A:4CA2 becomes FE80::2AA:FF:FE9A:4CA2. The double colon can only appear once in an address.

The last important part of the IPv6 address is the prefix. IPv6 address prefixes express a route, address space, or an additional range. IPv6 always uses the *ipv6 address/prefix-length* notation when describing IPv6 prefixes, similar to the subnet notation in IPv4 addresses, for example, an IPv6 prefix for subnets is 2001:DB8:0:3F3B::/64, and a prefix for routes is 2001:DB8:3F::/48.

IPv6 IP Address Types

There are three main types of IP addresses supported by IPv6. These three types are unicast addresses, anycast addresses, and multicast addresses. In addition, special address types, such as unspecified addresses, default routes, local addresses, and solicited-node mulitcast addresses are also supported. IPv6 does not support broadcast addresses because IPv6 architecture attempts to only contact the smallest number of nodes on the link in order to keep network disturbances to a minimum. The supported IPv6 address types are discussed in the following sections.

Unicast Addresses

IPv6 unicast addresses are addresses of a single interface and supports traffic delivery to a single node (1:1). Unicast addresses include global IPv6 addresses; locally used addresses, such as link-local addresses and site-local addresses; unique local addresses; and some special addresses.

Global unicast addresses have an address scope of the entire IPv6 Internet, and are basically the equivalent of IPv4 public addresses. These addresses are composed of three parts: a global routing prefix, a subnet ID, and an interface ID.

Link-local addresses have an address scope of a single link, and are the equivalent to automatic private IP addressing (APIPA) in IPv4. These addresses have a prefix of FE80::/64, and they are used for single subnets with routerless configurations and for ND processes. These addresses are mandatory for communication between two IPv6 devices, and are automatically assigned on an interface as soon as IPv6 is enabled. Link-local addresses also use zone IDs to identify specific links or sites. Link-local addresses typically set the zone ID to the interface index of the sending interface, and site-local addresses typically have a zone ID of 1 (unless multiple sites are used).

Unique-local addresses (ULAs) are IPv6 addresses that are private to an organization, yet are unique across all of the organization's sites. These addresses are not routable on the Internet, and typically have a prefix of FC00::/8. These addresses have a global scope, and therefore do not require zone IDs.

There are also special IPv6 addresses, which include unspecified addresses and loopback addresses. In IPv6, unspecified addresses take the form of 0:0:0:0:0:0:0:0 (or ::) and are the equivalent of 0.0.0.0 in IPv4. Loopback addresses in IPv6 take the form 0:0:0:0:0:0:0:1 (or ::1) and are the equivalent of 127.0.0.1 in IPv4.

Anycast Addresses

IPv6 anycast addresses are addresses assigned to multiple devices in the network. Anycast message delivery functions in a 1:any logical fashion, just as unicast message delivery is 1:1 and multicast message delivery is 1:many. Anycast packets are routed to the nearest anycast member by the network. Anycast addresses are indistinguishable from unicast addresses, but a device knows locally when an anycast address is used because the address is tied to the local application using the anycast address and the device does not test for uniqueness since more than one device can use the address.

Multicast Addresses

IPv6 multicast addresses are used as identifiers for a set of interfaces that belong to different nodes (1:many). These addresses are used to deliver packets to all interfaces identified by the multicast address. IPv6 multicast addresses use the prefix FF00::/8. Multicast addresses contain a group ID that identifies to which type of IPv6 devices the packets are being sent. [Table 4](#) describes the group IDs used in IPv6 multicast addresses.

Table 4. IPv6 Multicast Address Group IDs

Group ID	Group
01	Nodes
02	Link
05	Site

These group IDs are entered in the second 4 bits of the address. For example, an address with the prefix FF02::1 specifies that the packets are being sent to all nodes on the link.

In addition to IPv6 multicast addresses, there are also solicited-node multicast addresses. These addresses are analogous to IPv4 broadcast addresses since broadcast addresses are not supported by IPv6. This address is a multicast group that corresponds to an IPv6 unicast or anycast address. IPv6 solicited-node multicast addresses have the prefix FF02::1:FF00:/104. Solicited-node multicast addresses are derived

using the 24 low-order bits of corresponding IPv6 unicast or anycast addresses. For example, if an interface has an IPv6 address of FE80::2AA:FF:FE3F:2A1C, the corresponding solicited-node address is FF02::1:FF3F:2A1C (notice the lower 24 bits of the interface address, 3F:2A1C, become the solicited-node address).

Multicast Listener Discovery (MLD), a protocol used by IPv6 routers to discover multicast listeners, is used by AOS routers to respond to MLD queries and to send MLD reports. MLD messages are used by IPv6 protocols such as Open Shortest Path First version 3 (OSPFv3), Dynamic Host Control Protocol version 6 (DHCPv6), Virtual Router Redundancy Protocol version 3 (VRRPv3), and IPv6 network interfaces.

IPv6 and IPv4 Addresses

Table 5 outlines the differences between IP addressing in IPv4 and IPv6.

Table 5. IPv4 Addresses and IPv6 Equivalents

IPv4 Addresses	IPv6 Addresses
Internet address classes	N/A
Multicast addresses (224.0.0.0/4)	IPv6 multicast addresses (FF00::/8)
Broadcast addresses	Not supported
Unspecified address is 0.0.0.0	Unspecified address is ::
Loopback address is 127.0.0.1	Loopback address is ::1
Public IP addresses are used	Global unicast addresses are used
Private IP addresses are used	Unique-local addresses are used (FD00::/7 prefix)
APIPA addresses are used	Link-local addresses are used (FE80::/64 prefix)
Addresses expressed in dotted decimal notation	Addresses expressed in colon hexadecimal format
Subnet masks or prefix lengths are used	Only prefix lengths are used

IPv6 Route Cache and Traffic Processing

The IPv6 route cache is used to accelerate the packet forwarding procedure between interfaces in the AOS product particularly when stateful packet inspection (NAT, firewall, VPN, etc.) is not required. The route cache stores the results of the route table lookup when the first packet destined for a specific IPv6 address is forwarded, as well as storing the results of the ND address resolution process. The cache can be used as an interface lookup table whose entries use the IPv6 destination address as the key and contain the egress interface for the packet, the gateway IPv6 address used, the gateway MAC address used (if applicable), and a packet count revealing how many times the entry has been used. By storing this information, the packet processing speed increases because information for a particular packet flow to a particular IPv6 destination is readily accessible.

A route cache entry is uniquely identified by the destination IPv6 address of the packet. When a packet is received on an interface, the destination IPv6 address in the IP header of the packet is used to lookup an entry in the route cache. If the entry exists, then the associated information is used to accelerate the packet's forwarding out the egress interface. The accelerated forwarding speed of packets with a matching route cache entry is often referred to as the IPv6 fast path. If a packet arrives that does not have a matching route cache entry, the packet is forwarded through the traditional data path, which performs route table lookup and ND address resolution. As the packet passes through the regular IPv6 data path, a new route cache entry is created for the IPv6 destination, so that future packets for this destination are processed more quickly.

Route cache entries are added only after an IPv6 packet has traversed the regular IP data path. Link-local, unicast, and non-routed multicasted addresses are all eligible for entry in the route cache. Multicast routed addresses are not added to the cache. When a new entry would occupy the same location as a previously cached entry, the old entry is removed and replaced by the new entry.

Entries are removed from the route cache when the subsystem associated with the entry cause stale entries to be deleted. For example, when a forwarding route is changed or removed from the route table, all route cache entries that might have been covered by this route are cleared. In addition, whenever the default route is removed, all route cache entries in a given virtual private network (VPN) routing and forwarding (VRF) instance are removed. Entries are also removed whenever an ND cache entry is removed, because all route cache entries that use the ND cache entry's address as the IPv6 gateway address are cleared. Lastly, entries are removed when an interface changes states from active to inactive, and whenever IPv6 forwarding is globally disabled.



Some AOS features do not use the route cache. When these features are enabled, or when the IPv6 route cache is disabled, IPv6 packets are forwarded without the use of the route cache. For a list of these features, refer to [Hardware and Software Requirements and Limitations on page 10](#).

Router Processing in IPv6

The processing of packets and traffic flows in IPv6 differs slightly from that in IPv4. When routers are processing IPv6 packets, rarely used header fields are moved to separate options in extension headers. If the fields are not used at all, they are omitted, allowing faster packet processing. In addition, routers do not perform fragmenting in IPv6. Instead, IPv6 hosts are required to perform either path MTU (PMTU) discovery, perform end-to-end fragmentation (using the Fragment header), or to send packets no larger than the IPv6 default minimum MTU size of 1280 octets.

IPv6 headers are also not protected by a checksum as in IPv4. Error detection is assumed in IPv6 at the link layer or above, however, UDP checksum is required in IPv6. The disuse of checksum for the IPv6 packet eliminates the need for a router to recalculate a checksum when the header fields are changed, as it happens when the hop count diminishes.

MTU in IPv6

In IPv6, the minimum MTU is 1280 octets. Any link that has an MTU less than 1280 octets must use link fragmentation and reassembly that is transparent to IPv6 (for example, the Fragmentation header). Sources in the IPv6 network are expected to perform PMTU discovery to send packets larger than 1280 octets. PMTU works in the following manner: First, the sending node assumes the link MTU of the interface from

which the traffic is being forwarded and then sends the IPv6 packet at the link MTU size. If a router on the path is unable to forward the packet, it sends an ICMP *Packet Too Big* message back to the sending node containing the link MTU of the link on which the packet forwarding failed. The sending node then resets the PMTU to the value of the MTU field in the ICMPv6 *Packet Too Big* message, and the packet is resent.

Hardware and Software Requirements and Limitations

IPv6 is only supported in specific AOS devices using firmware 18.1.00 or later. For a complete listing of products that support IPv6, refer to the *Product Feature Matrix*, available online at <https://supportforums.adtran.com>.

AOS implements a dual IP stack method for supporting IPv4 and IPv6. Not all AOS features are compatible with IPv6. Refer to the *Product Feature Matrix*, available online at <https://supportforums.adtran.com>, for more information about the supported IPv6 features in AOS.

In AOS firmware release 18.3.01, the following IPv6 features were added:

- DNS for IPv6
- DHCP Server capability for IPv6
- HTTP(S) for IPv6
- IPv6 Host Mode for AOS Switches

In AOS firmware release R10.1.0, the following IPv6 features were added:

- IPv6 FTP ALG
- IPv6 Firewall Filtering
- IPv4 GRE Tunnel with IPv6 Payload
- IPv6 Quality of Service (QoS)

In AOS firmware release R10.3.0, the following IPv6 features were added:

- Network Time Protocol (NTP) for IPv6

In AOS firmware release R10.4.0, the following IPv6 features were added:

- IPv6 Rapid Route Engine

In AOS firmware release R10.5.0, the following IPv6 features were added:

- Open Shortest Path First version 3 (OSPFv3) for IPv6
- IPv6 support on the loopback interface

In AOS firmware release R10.7.0, the following IPv6 features were added:

- IPv6 IPsec

In AOS firmware release R10.8.0, the following IPv6 features were added:

- IPv6 support for Session Initiation Protocol (SIP) and Realtime Transfer Protocol (RTP). The specifics of IPv6 SIP operation in AOS are covered in the configuration guide *IPv6 SIP in AOS*, available online at <http://supportforums.adtran.com>.

In AOS firmware release R10.9.0, the following IPv6 features were added:

- DHCP client capability for IPv6
- Support for delegated and manually configured IPv6 named prefix variables.

In AOS firmware release R11.1.0, the following IPv6 features were added:

- DHCPv6 prefix delegation
- DHCPv6 rapid commit

In AOS firmware release R11.4.0, the following IPv6 features were added:

- MLD listener capability

In AOS firmware release R11.10.0, the following IPv6 features were added:

- RapidRoute Flow Bundling and Service Assurance
- Ability to specify how many incomplete entries are stored in the ND neighbor cache

In AOS firmware release R11.10.2, the following IPv6 features were added:

- Ability to match or block non-initial fragments in ACL entries.

Additional information about IPv6 concepts and usage, as well as other AOS IPv6 features (such as DHCPv6, IPv6 BGP, and IPv6 QoS) is outlined in [Additional Resources on page 79](#).

IPv6 Route Cache Limitations

The IPv6 route cache is not used by all AOS features. When certain features are enabled, or when the route cache is disabled, the route cache is not used, even if applicable entries in the cache exist for the packet. The following features affect the use of the IPv6 route cache:

- When unicast routing is disabled, then the IPv6 route cache and fast-path processing is disabled.
- If the route cache is disabled on a specific ingress interface, then IPv6 fast-path processing for packets received on that interface is disabled.
- When an inbound ACL is applied to an interface, the IPv6 fast-path processing is disabled for that interface.

In addition, some features limit which route cache entries can be created. The following features affect the creation of new IPv6 route cache entries:

- When the route cache is disabled on a specific interface, then new IPv6 route cache entries are not created.
- When the ingress or egress IPv6 interface is inactive (whether for signaling protocol failures, severing of the physical connection, or a manual shutdown), new IPv6 route cache entries are not created.
- New entries are not created when the route table lookup fails to find a valid egress interface.
- If unicast routing is globally disabled, no new route cache entries are created.
- If the ND address resolution process fails to find a valid link-layer address (MAC address), then no new route cache entries are created.
- If an ACL is applied to the ingress or egress interface, then no new route cache entries are created.

IPv6 Host Mode for AOS Switches

In AOS firmware release 18.3.01, AOS switches were fitted with the capability to act as a host in an IPv6 network. This feature allows switches to be managed using an IPv6 address. The following IPv6 features are now available on AOS switches:

- IPv6 Telnet Server
- IPv6 SSH Server
- IPv6 SNMP Server
- IPv6 HTTP(S) Server
- IPv6 TFTP Server
- IPv6 Ping and Traceroute features
- IPv6 DNS Client and Proxy features

Although these features are available on AOS switches, not all IPv6 functionality is available. Each switch operates in host only mode, and does not have the same capability for IPv6 security features as do AOS routers. VRF parameters are also not available on AOS switches. The switches can, however, use IPv6 neighbor discovery, stateless address auto-configuration, default address selection, etc. By default, any advertised routes in the switch will be given a default administrative distance of **2**, and switches will react just like AOS routers to router advertisement flags.

IPv6 Implementation in AOS

AOS implements IPv6 by incorporating two IP stacks into the AOS architecture. The dual-stack structure creates one stack for IPv4 use, and one stack for IPv6 use, allowing single feature upgrades from IPv4 to IPv6. In addition, using the dual-stack approach provides a method of transitioning from IPv4 to IPv6 on the network. The dual-stack functionality operates in one of three modes:

- The IPv4 stack is enabled and the IPv6 stack is disabled.
- The IPv6 stack is enabled and the IPv4 stack is disabled.
- Both stacks are enabled.

Both stacks can be operated independently, or they can operate in parallel. In its initial implementation, IPv6 is most often used for routing, while IPv4 is still used for network services and management.

[Figure 2](#) illustrates the dual-stack functionality in AOS.

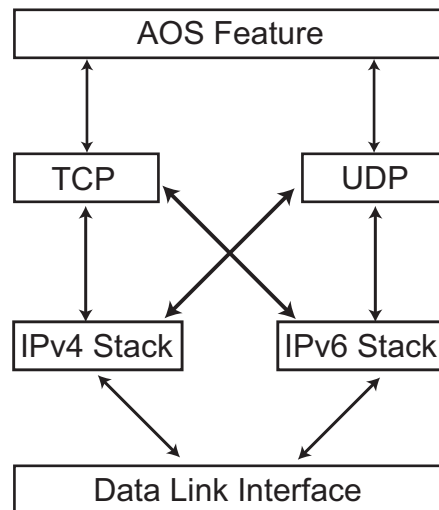


Figure 2. Dual-Stack IPv6 Implementation in AOS

Enabling IPv6 on an Interface in AOS

Because AOS uses the dual-stack for IPv6 implementation, the IPv6 stack must be enabled for the supported IPv6 features to be used. Enabling IPv6 on an interface in AOS is completed by using an IPv6 address or using the **ipv6** keyword with specific commands. For example, to enable IPv6 on an interface and cause the interface to join the link scoped all-nodes and all-routers multicast group, enter an IPv6 address on the interface.

IPv6 and the Interface in AOS

As with other AOS features, there are also additional IPv6 configuration commands for interfaces. These commands include MTU settings, IPv6 addressing, enabling IPv6, and enabling the IPv6 route cache. These IPv6 features for interfaces are described in the following sections.

General IPv6 MTU Settings

The MTU for IPv6 packets can be set on a per-interface basis. There are two methods for setting IPv6 MTUs for interfaces if required: one for Layer 3 interfaces, and one for the underlying Layer 1 and Layer 2 interfaces. For all interface types, use the **ipv6 mtu <size>** command to specify the IPv6 MTU in bytes from the interface's configuration mode. The minimum MTU setting for IPv6 is **1280** bytes, and the maximum is **1592** bytes. The IPv6 MTU value is independent of the IPv4 MTU setting. To set the IPv6 MTU value for an interface, enter the command from the interface's configuration mode as follows:

```

(config)#interface ppp 1
(config-ppp 1)#ipv6 mtu 1300
(config-ppp 1)#interface ethernet 0/1
(config-eth 0/1)#ipv6 mtu 1350
  
```

When the interface is forwarding the IPv6 packet as a router, if the packet size exceeds the IPv6 MTU of the egress interface, the packet is dropped and an ICMPv6 *Packet Too Big* message is sent to the source. When originating an IPv6 packet from the local IPv6 stack, and the packet is larger than the IPv6 MTU of the egress interface, the packet is fragmented and sent.

Applying Unicast IPv6 Addresses to an Interface

The following commands are used on a per-interface basis to apply unicast IPv6 addresses to the interface. You can manually enter an IPv6 address or you can automatically configure IPv6 addresses on the interface.

1. Use the **ipv6 address** *<ipv6 address/prefix-length>* command to assign a unicast IPv6 address to the interface and enable IPv6 processing on the interface. This address is manually configured by specifying the IPv6 address, including all prefix and host bits. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X/<Z>**), for example, **2001:DB8::1/64**. The prefix length (**<Z>**) is an integer with a value between **0** and **128**. This address can be a global unicast address or a unique local address, but not a link-local address. By default, no IPv6 address is configured on the interface and IPv6 processing is not enabled on the interface. Using the **no** form of this command with a specified IPv6 address removes that IPv6 address from the interface. Using the **no** form of this command without a specified IPv6 address removes all manually configured IPv6 addresses from the interface. To add a unicast IPv6 address to an interface and enable IPv6 processing on the interface, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 address 2001:DB8::1/64
```

2. Use the **ipv6 address** *<ipv6 prefix/prefix-length>* **eui-64** command to assign a unicast IPv6 address and enable IPv6 processing on the interface. The **eui-64** parameter specifies that the IPv6 address is constructed using the specified prefix in the high-order bits and followed by the EUI-64 interface ID in the lower 64 bits. The EUI-64 interface ID is automatically placed in the IPv6 address. Any manually configured bits beyond the address's prefix length are set to **0**; however, any manually configured bits within the prefix length that extend into the lower 64 bits take precedence over the interface ID bits. The prefix value is specified in colon hexadecimal format (**X: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 resulting IPv6 address may be a global unicast address or a unique local address, but cannot be a link-local address. By default, no IPv6 addresses are configured on the interface and IPv6 processing is not enabled. Using the **no** form of this command with a specified IPv6 address removes that IPv6 address from the interface. Using the **no** form of this command without a specified IPv6 address removes all manually configured IPv6 addresses from the interface. To manually enter an IPv6 unicast address created with an EUI-64 interface ID on the interface and enable IPv6 processing, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 address 2001:DB8:3F::/64 eui-64
```

3. Use the **ipv6 address** *<ipv6 address>* **link-local** command to assign a link-local IPv6 address to the interface and enable IPv6 processing on the interface. An interface automatically is assigned a default link-local address, but this address may be reconfigured to be a different value. The lower 64 bits of the specified address become the interface ID, overriding the default interface ID. Any other address that uses the EUI-64 parameter to automatically place the interface ID in the lower 64 bits of the IPv6 address use the new value for the interface ID. The *<ipv6 address>* for a link-local IPv6 address is

specified in the format **FE80::<bits>**. The <bits> are the lower 64 bits of the link-local IPv6 address, and since link-local addresses have no prefix, the bits entered form the entire IPv6 address. These bits also become the new interface ID, overriding the default interface ID. The **link-local** parameter specifies this is a manually configured link-local address. Any manually configured link-local address will replace an automatically configured link-local address for the interface. Using the **no** form of this command with a specified IPv6 address removes that IPv6 address from the interface. Using the **no** form of this command without a specified IPv6 address, removes all manually configured IPv6 addresses from the interface. To manually enter a link-local IPv6 address on the interface and enable IPv6 processing, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 address fe80::220:8FF:FE54:F9D8 link-local
```



*If IPv6 is enabled on the interface using a command other than **link-local**, a link-local address will automatically be created based on the interface's ID. The interface ID is based on the MAC address for an Ethernet interface and based on the system MAC address (usually the first Ethernet interface in the unit) for PPP interfaces.*

- Use the **ipv6 address autoconfig [default] [metric <value>]** command to enable IPv6 processing on the interface, create a link-local IPv6 address for the interface, and allow the interface to automatically configure itself based on advertisements from other routers on the link. When autoconfiguration is enabled, the interface listens for RA messages that tell the interface how it should be configured. The interface then creates addresses for advertised 64-bit prefixes with the A flag set using stateless address autoconfiguration (SLAAC). The addresses use the EUI-64 interface ID in the lower 64 bits of the address. A route type of *Connected* is added to the route table if the L flag on the prefix advertisement (on-link flag) is also set. The optional **default** parameter specifies that the interface maintain a list of advertising routers that are willing to be default routers. You can also optionally specify the administrative distance for a default router maintained in the default router list by entering the **metric <value>** parameter. The <value> is the administrative distance, and has a range of **1** to **255**. Low administrative distances are preferred. By default, the default router administrative distance is set to **2**. Using the **no** form of this command removes all automatically configured addresses, prefixes, and any resulting routes from the interface and also causes the interface to cease processing received RAs. To enable IPv6 processing, create a link-local address, and allow the interface to automatically configure itself, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 address autoconfig
```

- Use the **ipv6** command to enable IPv6 processing and create a link-local address on an interface when other IPv6 unicast addresses are not needed on the interface. This command is not necessary nor effectual when any of the other IPv6 address commands have been issued on the interface. By default, IPv6 is not enabled on the interface. Using the **no** form of this command disables all IPv6 processing on the interface and removes all IPv6 configurations from the interface. To enable IPv6 and create a link-local IPv6 address on the interface (when no other unicast IPv6 address is needed on the interface), enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6
```

Configuring the IPv6 Route Cache on an Interface

The IPv6 route cache is enabled on a per-interface basis. To enable the IPv6 route cache on an interface, enter the **ipv6 route-cache** command from the interface's configuration mode. Use the **no** form of this command to disable the route cache. By default, the route cache is enabled.

To enable the route cache after it has been disabled, enter the command as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 route-cache
```

Configuring IPv6 Routing in AOS

IPv6 routing in AOS supports unicast routing, adding static routes to the IPv6 route table, and optionally entering static entries into the neighbor cache. These features and their associated commands are discussed in the following sections.

IPv6 Unicast Routing

IPv6 unicast routing is enabled in AOS using the **ipv6 unicast-routing** command from the Global Configuration mode prompt. This command functions similarly to the **ip routing** command for IPv4 services. In order to enable IPv6 unicast routing, you must first configure interfaces to use IPv6 before IPv6 communication takes place. Refer to *IPv6 and the Interface in AOS on page 13* for more information about configuring the IPv6 interface. When IPv6 unicast routing is enabled globally, the router flag is set to **1** in neighbor advertise messages.

Using the **no** form of this command disables the IPv6 routing subsystem, removes any routing protocol entries from the IPv6 route table, causes IPv6 routing functions to cease, and disables IPv6 unicast routing. In addition, NA messages are sent at each interface indicating the neighbor is no longer a router (router flag is set to **0**), and that the router is no longer the default router for any advertised prefixes. When IPv6 unicast routing is disabled, the existing IPv6 configuration is retained, but no IPv6 packets are routed and no routing resources are consumed.

If IPv6 unicast routing is not enabled, but an interface has IPv6 enabled, that interface may communicate as an IPv6 host to other devices. If IPv6 packets are received that are not addressed to that interface, the packets are dropped.

To enable IPv6 unicast routing, and specify the router as an IPv6 neighbor, enter the command from the Global Configuration mode as follows:

```
(config)#ipv6 unicast-routing
```

Adding Static Routes to the IPv6 Route Table

Static routes can be added to the IPv6 route table using a single command from the Global Configuration mode prompt. By default, no static routes exist in the IPv6 route table. Each static route is only added to the IPv6 route table when the IPv6 interface is configured and in an UP state. There are three types of static routes that can be used: directly attached, recursive, and fully specified.

A directly attached static route is a route in which the next hop for the route is entered as an interface. Packets destined for the specified network are assumed to be directly reachable on the specified interface. If you are using a directly attached static route, and the interface you are using uses Layer 2 addresses (for example, as an Ethernet interface does), then address resolution is performed when a packet is delivered to the network. For PPP interfaces, the packet is simply forwarded through the interface in the same way that a packet is forwarded when an IPv6 on-link prefix is defined at the interface.

A recursive static route is a route in which the next hop for the route is entered as the IPv6 address of the next-hop router. When a recursive static route is used, AOS attempts to determine the interface used to reach the next-hop address. Recursive routes are added to the route table only when the router has determined which interface to use for egress traffic.

A fully specified static route is a route in which the next hop is entered as an IPv6 address and an interface for the next-hop router is specified. This type of static route restricts the use of the route to the specified interface. A fully specified static route **MUST** be used when the next hop is specified by its link-local address, which alone has no context of location.

To add a static route to the IPv6 routing table, enter the **ipv6 route** [**vrf** <name>] <ipv6 prefix/prefix-length> <ipv6 address> [<interface> | **null 0**] [<administrative distance>] [**tag** <value>] command from the Global Configuration mode prompt. The optional **vrf** <name> parameter of the command specifies a nondefault VLAN routing and forwarding (VRF) interface on which to create the static route. Using the **no** form of this command removes the static route from the IPv6 routing table.

The <ipv6 prefix/prefix-length> parameter specifies the network defined by this static route entry. 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 IPv6 prefix cannot be a link-local address.

The <ipv6 address> parameter specifies the next-hop IPv6 address (other than the link-local address) defined by the static route. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. If the next-hop device does not share a link with this router, recursive lookups are performed to locate the directly connected next hop. To use a local-link address as the next hop, use the <ipv6 address> parameter in conjunction with [<interface> | **null 0**] parameter (making this a fully specified static route).

The [<interface> | **null 0**] parameter optionally specifies an interface on the router which connects to the next-hop IPv6 device on the path toward the specified network. When the next-hop is reached by a PPP link, this router's egress interface can be specified instead of a next-hop address. When specifying a next-hop address that is a link-local address, both the next hop address and the connecting interface **MUST** be specified. Interfaces are specified in the <interface> <slot/port | interface id> format. For a list of applicable egress interfaces, enter **ipv6 route** <ipv6 prefix/prefix-length> <ipv6 address> ?.

The optional <administrative distance> parameter of this command specifies an administrative distance associated with the static route, and is used to determine the best route when multiple routes to the same destination exist. The route with the lowest administrative distance is the preferred route. Administrative distance has a default value of **1** and a range of **1** to **255**.

The optional **tag** <value> parameter of this command specifies a number to use as a tag for this route. Valid tag range is **1** to **65535**.

For example, to enter a static route in the IPv6 routing table with a local-link next hop address, egress from the **ethernet 0/1** interface, a tag of **3**, and an administrative distance of **2**, enter the command as follows:

```
(config)#ipv6 route 2001:db8:3f::/48 fe80::202:b3ff:fe1e:8345 ethernet 0/1 tag 3 2
```

Creating a Static Entry in the Neighbor Cache

In IPv6, neighbors are usually managed dynamically using the ND protocol. However, you can manually enter a static entry to the neighbor cache using the **ipv6 neighbor** *<ipv6 address>* *<interface>* *<mac address>* command. This command functions similarly to the **arp** command for adding a static ARP entry to the IPv4 ARP cache.

The *<ipv6 address>* parameter of the **ipv6 neighbor** command specifies the neighbor's IPv6 address. IPv6 addresses are entered in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The *<interface>* parameter of the command specifies the interface to the link on which this neighbor is connected. Interfaces are specified using the *<interface>* *<slot/port | interface id>* format, for example, for an Ethernet interface, you would specify **ethernet 0/1**. The *<mac address>* parameter of the command specifies the neighbor's MAC address. MAC addresses should be expressed in the following format: **XX:XX:XX:XX:XX:XX**, for example, **00:A0:C8:00:00:01**.

By default, no static neighbor entries exist in the neighbor cache. Using the **no** form of the **ipv6 neighbor** command removes the static entry from the neighbor cache.

When you enter a static entry into the neighbor cache, you should be aware of the following:

- A static entry entirely overrides an existing or new dynamic entry learned through neighbor discovery.
- NUD is not performed on static neighbors, so the neighbor's state is limited to either an incomplete modified state (interface is down) or a reachable modified state (interface is up).
- Using the **no** form of the **ipv6 neighbor** command removes static entries and not dynamic entries. Using the **clear ipv6 neighbor** command clears the dynamic entries and not the static entries.
- Disabling IPv6 on an interface does not remove the static neighbor cache entries, although it will change the entry states to incomplete.

To add a static entry to the neighbor cache, enter the command from the Global Configuration mode prompt as follows:

```
(config)#ipv6 neighbor 2001:DB8:3F::1 ethernet 0/1 00:A0:C8:00:00:01
```

IPv6 Load Sharing

IPv6 load sharing allows parallel routes in the IPv6 route table to be used to balance IPv6 traffic to a specific destination across up to six equal paths. To enable IPv6 load sharing, enter the **ipv6 load-sharing** [**per-destination** | **per-packet**] command from the Global Configuration mode. When this command is enabled, the IPv6 route table can use multiple "best" routes and alternate between them. When this command is disabled, the IPv6 route table uses a single "best" route. The **per-destination** parameter of this command specifies that the route used to forward a packet is based on a hash of the source and destination IPv6 address in the packet. The **per-packet** parameter of this command specifies that each forwarding route lookup rotates through all the parallel "best" routes. Using the **no** form of this command disables the load sharing feature. By default, load sharing is disabled. To enable IPv6 load sharing, enter the command from the Global Configuration mode as follows:

```
(config)#ipv6 load-sharing per-destination
```

IPv6 IPsec

IPv6 IPsec is the newest version of Internet Protocol security that supports IPv6 functionality. In AOS, IPsec provides routers with site-to-site connectivity over an untrusted IPv6 network, secured communication that terminates on the router, and authentication and encryption for OSPFv3. *Figure 3* illustrates IPv6 IPsec function in the network. In this illustration, a host on the 2001:DB8:1::/64 network communicates with a host on the 2001:DB8:2::/64 network and the IPv6 packets are encapsulated into an IPv6 IPsec tunnel. Traffic is passed between the network-facing IPv6 addresses of the two tunnel endpoint routers (2001:DB8:3::1 and 2001:DB8:4::1). The terminating tunnel endpoint removes the IPv6 tunnel headers and processes the payload, revealing the original IPv6 packet before sending it to the destination host.

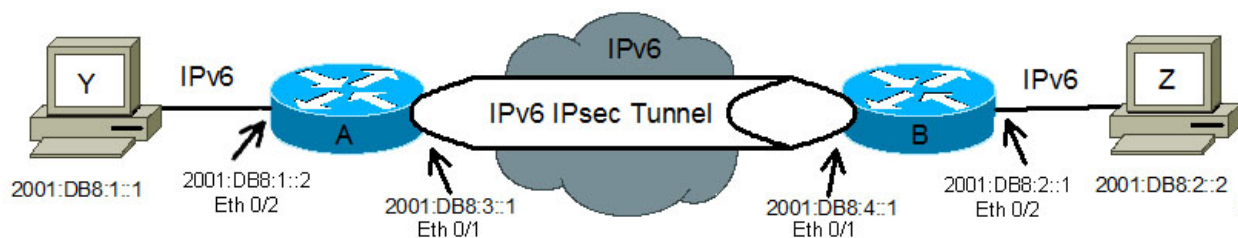


Figure 3. IPv6 over IPv6 IPsec in Tunnel Mode

IPv6 IPsec Functionality

IPv6 IPsec functions in the same manner as IPv4 IPsec. It uses ACLs to identify traffic that is to be protected, and then uses a crypto map, configured with the proper IPv6 IPsec settings, to reference the ACL. The crypto map is then applied as a policy to an interface. Packets traversing the interface that match the crypto map policy are processed following the settings outlined in the crypto map (protecting packets or dropping them). Packets that do not match the applied crypto map are sent or received in an unprotected state. To implement IPv6 IPsec, you will create an IPv6 crypto map, just as with IPv4 IPsec, except that the IPv6 crypto map uses IPv6 settings. When the crypto map policy is applied to an interface, it does not create a new interface (just as with IPv4 IPsec).

Because ACL-based IPv6 IPsec tunnels are not actual IP interfaces, the typical functions of an IPv6 interface do not take place over the IPv6 IPsec tunnel. This includes any kind of IPv6 address or route derived from a tunnel interface, IP multicast, DAD, NUD, and SLAAC functions of Neighbor Discovery, ICMPv6 error generation and handling, and DHCPv6. All IPv6 operation of the interface to which the tunnel is applied, however, will function normally.

IPv6 IPsec Considerations

There are many details to consider when configuring IPv6 IPsec. The following sections outline specific topics for IPv6 IPsec.

ICMPv6 Errors

ICMP errors destined to the router are examined to determine whether they should have received IPsec protection. If IPsec protection was not provided correctly, the error is dropped and not processed further.

Off-SA conditions that require the router to send back an ICMPv6 error are sent unprotected. On-SA conditions that require the router to send back an ICMPv6 error are sent with IPsec protection.

IPv6 IPsec and Neighbor Discovery

The application of an IPv6 crypto map to an interface does not interfere with the ND functions of the interface. In addition, since an ACL-based crypto map does not create an interface instance, ND does not take place over the IPsec SA, even when in tunnel mode.

EUI-64 Formatted Addresses

Since an ACL-based crypto map is not a formal interface construct, there is no interface ID or addresses other than those of the interface to which the crypto map is applied.

IPv6 Source Address Validation


As with IPv4 IPsec, when IPv6 IPsec is enabled using the **ipv6 crypto** command, the IPv6 firewall is automatically invoked. Stateless packet processing is employed. To perform stateful processing, the IPv6 firewall must be enabled using the **ipv6 firewall** command. Using IPv6 IPsec for OSPFv3 protection does not automatically invoke the IPv6 firewall.

The IPv6 firewall will perform address validation on IPv6 packets that are sent in tunnel mode. If the original packets have any of the following source addresses, they will be silently discarded:

- Multicast addresses (FF00::/8)
- The loopback address (::1)
- IPv4-mapped IPv6 addresses (::FFFF:0:0/96)
- IPv4-compatible IPv6 addresses (::/96) excluding the unspecified address (::/128)

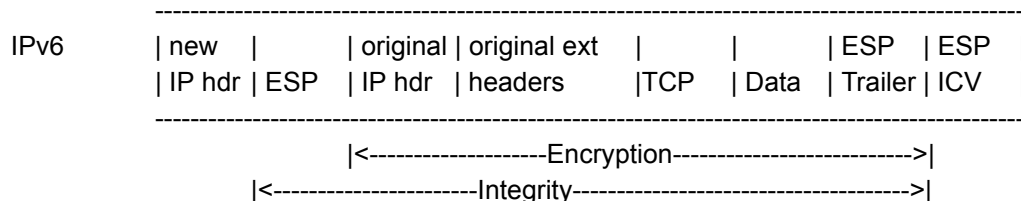
IPv6 IPsec Packet Format

When using ESP, an IPv6 over IPv6 IPsec tunnel mode packet has the following format:

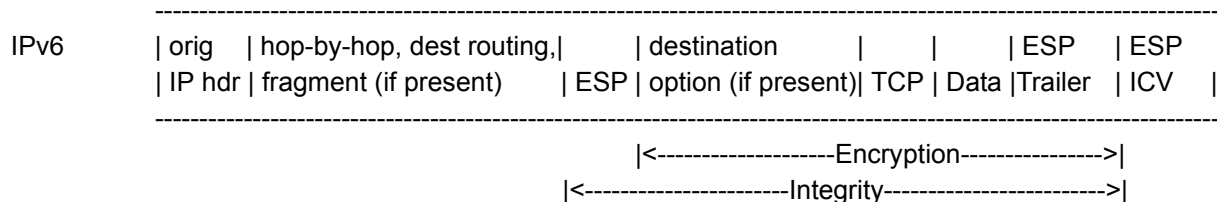


NOTE

The examples below assume a TCP payload; however, other payload types can also be used.



When using ESP, an IPv6 over IPv6 IPsec transport mode packet has the following format:



Hop Limit

A tunnel mode IPv6 IPsec SA functions like a point-to-point link for the original IPv6 traffic protected by the SA. The SA endpoints of a tunnel connection decrement the IPv6 hop limit of the original packet by 1 before forwarding. The sending tunnel endpoint sets the IPv6 hop limit field in the outer IPv6 header to 255.

Traffic Class

In IPv6 IPsec tunnel mode, the traffic class value from the inner IPv6 header is copied to the outer IPv6 header when the packet is encapsulated. The IPv6 traffic class value is ignored when the packet is decapsulated, which leaves the IPv6 header with the original traffic class value. This behavior is similar to the IPv4 IPsec function.

IPv6 IPsec Key Management

In the initial implementation of IPv6 IPsec (AOS firmware release R10.7.0), only manual keys are supported for IPv6 IPsec configuration. The manual keys supported are identical to those used by IPv4 IPsec. These keys are specified as hex characters, and their sizes for each algorithm are outlined in [Table 6](#).

Table 6. IPsec Encryption/Authentication Key Types and Sizes

Key Type	Size (bytes)
3DES	24
AES-CBC 128	16
AES-CBC 192	24
AES-CBC 256	32
DES	8
MD5	16
SHA1	20

VRF Support

IPv6 IPsec is a VRF-aware feature, meaning that IPsec using ACL-based crypto maps can be configured at the global route table and within individual VRF instances. A crypto map can be configured to protect traffic to or from a single VRF. Tunnel mode IPsec carries traffic from that VRF using a local tunnel endpoint that is in the same VRF, and a remote tunnel endpoint that can be reached from that VRF. All

configuration is done in the default VRF on the interface, unless a nondefault VRF is specified in the configuration. When a crypto map is assigned to an interface, that instance of the crypto map becomes associated with the VRF to which the interface is assigned. The SPI database is independent for each VRF. SPIs must be unique within a VRF, but can be duplicated across VRFs and address families.

Using IPv4 and IPv6 IPsec Concurrently

With IPv4 IPsec, the router can establish multiple IPsec connections (SA pairs) to other devices. Connections can be derived from multiple entries in a single crypto map, or from different crypto maps applied to different interfaces. The same is true of IPv6 IPsec. Multiple connections can be made with either or both IP versions.

In AOS, there is a limit to the number of IPsec SAs that are supported on each AOS device. With the implementation of IPv6 IPsec, that limit is shared between IPv4 and IPv6 IPsec. The shared space can be all IPv4 SAs, all IPv6 SAs, all SAs for OSPF, or any combination thereof.

OSPFv3 and IPv6 IPsec

In OSPFv3 for IPv6, authentication has been removed from the OSPF protocol. When running over IPv6, OSPF relies on the IP Encapsulating Security Payload (ESP) and IP Authentication Header (AH) protocols to ensure integrity and authentication or confidentiality of routing exchanges (refer to RFC 4552). Protection of OSPF packet exchanges against accidental data corruption is provided by the standard IPv6 upper layer checksum, which covers the entire OSPF packet and the prepended IPv6 pseudo-header.

In addition, because routers act as hosts when performing OSPF, they are required to support IPsec transport modes. Therefore, support for ESP is included in OSPFv3, and AH is allowed. Authentication is required using either ESP or AH, while encryption is optional. However, only ciphers suitable for manual keys are allowed.

In AOS, OSPFv3 authentication is completed using IPsec. IPsec protection of OSPF can be configured at the interface or OSPF area level. Area-level authentication eases configuration management when the same security is to be applied to multiple interfaces. When protection is specified at the interface level, the configured security parameter index (SPI) is used only at that interface. When protection is specified at the area level, the SPI is used at each interface in that area.

To balance management advantages of area-level protection with concerns over interoperability and consistency when using multiple OSPFv3 instances at an interface, AOS supports protection at the area level, along with a set of rules to choose one set of security associations (SAs) for an interface when multiple protections are configured. When multiple OSPFv3 protections are specified that affect the same interface, one set is selected to protect all OSPFv3 instances on that interface. The set is chosen using the following rules processed in this order:

- The protection configured at the interface level (including null), then everything else
- The protection configured at the IPv6 address family instance, then everything else
- The protection configured at the IPv4 address family instance, then everything else (IPv4 address family is not supported as of AOS firmware release R10.5.0)
- No protection

When OSPFv3 authentication is enabled, received OSPFv3 packets that are not protected with AH or ESP, or packets that fail authentication checks, are discarded. Similarly, when OSPFv3 confidentiality is enabled, OSPFv3 packets that are not protected with ESP, or that fail the confidentiality checks, are discarded.

In AOS, IPsec uses the IP header to distinguish between IPv4 and IPv6 OSPF packets. Each AOS product that supports IPsec has a limited number of IPsec SAs it can support. In previous firmware versions, that limit consisted of only IPv4 IPsec SAs. With the addition of IPv6 IPsec and IPsec protection of OSPFv3, that limit is now a shared space of SA types. The space can be consumed by all IPv4 SAs, all IPv6 SAs, all SAs for OSPF, or any other combination. You should keep this in mind when configuring IPsec SAs for your networking applications.

In the AOS CLI, IPsec encryption and authentication algorithm keys are used in OSPFv3 commands. These keys are specified as hex characters, and their sizes for each algorithm are outlined in [Table 7](#).

Table 7. IPsec Encryption/Authentication Key Types and Sizes

Key Type	Size (bytes)
3DES	48
AES-CBC 128	32
AES-CBC 192	48
AES-CBC 256	64
DES	16
MD5	32
SHA1	40

Configuring IPv6 IPsec in AOS

IPv6 IPsec is configured in AOS using the CLI. The commands for IPv6 IPsec are separated into three main categories: global IPv6 IPsec settings, crypto map settings, and interface IPv6 IPsec settings. The global IPv6 IPsec settings are used to enable IPv6 IPsec, define the transform configuration for securing data, and to create a crypto map entry (and enter the entry's configuration mode). The IPv6 IPsec crypto map settings are used to designate the IPv6 traffic to be protected (using IPv6 ACLs), specify the peer device for IPv6 IPsec tunnel mode, define the encryption and authentication keys for the map, and to assign a transform set to the map. The IPv6 IPsec interface settings are used to apply the crypto map to the interface. The following sections describe the commands used in IPv6 IPsec configuration.

IPv6 IPsec Global Configuration

To configure the global parameters for IPv6 IPsec in AOS, use the commands described in this section.

1. To enable IPv6 IPsec, enter the **ipv6 crypto [vrf <name>]** command from the Global Configuration mode. By default, IPv6 IPsec is not enabled. You can optionally specify a nondefault VRF instance on which to enable IPv6 IPsec using the optional **vrf <name>** parameter. If a VRF is not specified, IPv6

IPsec is enabled on the default VRF. Use the **no** form of this command to disable IPv6 IPsec. To enable IPv6 IPsec on the default VRF instance, enter the command as follows:

```
(config)#ipv6 crypto
(config)#
```



*The **no** form of this command disables IPv6 IPsec, but leaves all IPsec settings in place.*

- Use the **ipv6 crypto ipsec transform-set** *<name>* *<parameters>* to define an IPv6 transform set. Transform sets are used to define the configuration for securing data. They are then applied to crypto map entries, which reference them for specific security algorithms. If the transform set is deleted, any references to the transform set by other functions are removed, leaving them incomplete. The *<name>* parameter specifies the name of the transform set you are defining. This name must be unique among IPv6 transform sets. Entering the name of an existing transform set re-enters that transform set's configuration mode. Names are specified in an alphanumeric string of up to 80 characters. The *<parameters>* keyword assigns a combination of the following:
 - zero or one AH authentication algorithms
 - zero or one ESP encryption algorithms
 - zero or one ESP authentication algorithm

In addition, if **esp-null** is used for encryption, an ESP authentication algorithm must be specified; it is not optional in this case. Available security algorithms are outlined in the following tables:

Table 8. AH Authentication Algorithms

Algorithm Keyword	Description
ah-md5-hmac	Authentication Header. Uses 16-byte key and HMAC-MD5-96 authentication.
ah-sha-hmac	Authentication Header. Uses 20-byte key and HMAC-SHA1-96 authentication.

Table 9. ESP Encryption Algorithms

Algorithm Keyword	Description
esp-des	Encapsulating Security Payload. Data encryption standard using cipher block chaining and an 8-byte key (DES-56-CBC).
esp-3des	Encapsulating Security Payload. Data encryption standard using cipher block chaining and a 24-byte key (3DES-168-CBC).
esp-aes-128-cbc	Encapsulating Security Payload. Advanced encryption standard using cipher block chaining and a 16-byte key.

Table 9. ESP Encryption Algorithms (Continued)

Algorithm Keyword	Description
esp-aes-192-cbc	Encapsulating Security Payload. Advanced encryption standard using cipher block chaining and a 24-byte key.
esp-aes-256-cbc	Encapsulating Security Payload. Advanced encryption standard using cipher block chaining and a 32-byte key.
esp-null	Encapsulating Security Payload with no encryption.

Table 10. ESP Authentication Algorithms

Algorithm Keyword	Description
esp-md5-hmac	Encapsulating Security Payload. Uses 16-byte key and HMAC-MD5-96 authentication.
esp-sha-hmac	Encapsulating Security Payload. Uses 20-byte key and HMAC-SHA1-96 authentication.

By default, no IPv6 IPsec transform sets are configured. Using the **no** form of this command removes the transform set. The following example creates a transform set named **SET1** that specifies ESP with 3DES encryption and SHA1 authentication:

```
(config)#ipv6 crypto ipsec transform-set SET1 esp-3des esp-sha-hmac
```

The transform set can then be applied to the crypto map entry when the entry is created (refer to [IPv6 IPsec Crypto Map Configuration on page 26](#)).

- The last step in global configuration of IPv6 IPsec is to create a crypto map entry and enter the crypto map entry's configuration mode. Use the **ipv6 crypto map <name> <index> ipsec-manual** command to create a manually keyed IPv6 crypto map entry (if none exists with that name) and to enter the manually keyed crypto map entry configuration mode. If a crypto map entry with that name and index already exists, you can enter the command as **ipv6 crypto map <name> <index>** and enter the entry's configuration mode. The *<index>* parameter assigns a crypto map sequence number. Valid range is **0** to **65535**. The **ipsec-manual** specifies that the keys are supplied manually (rather than negotiated with IKE). By default, no IPv6 crypto maps exist. Using the **no** form of the **ipv6 crypto map <name> <index>** command removes the crypto map entry. If that entry was the last entry in the map, the map is removed from the interface. Using the **no** form of the **ipv6 crypto map <name>** command without the *<index>* parameter removes the map and all of its settings. If the map is also used on an interface, removing the map also removes the map on any interfaces to which it is assigned. To create a manually-keyed IPv6 crypto map, enter the command from the Global Configuration mode as follows:

```
(config)#ipv6 crypto map VPN 10 ipsec-manual  
(config-crypto6-map)#
```

- You can optionally choose to disable the reverse path forwarding (RPF) check on the crypto map using the **no ipv6 crypto map <name> rpf-check** command. By default, RPF checking is enabled. When enabled, RPF checking of tunnel traffic disallows tunnel traffic that is spoofed from another tunnel. This

check is applied to any SA created from any entry in the named crypto map. The *<name>* parameter specifies the map on which RPF checking should be enabled or disabled. Using the **no** form of this command disables RPF checking. To disable RPF checking for a specific crypto map, enter the command from the Global Configuration mode as follows:

```
(config)#no ipv6 crypto map MAP1 rpf-check
(config)#
```

IPv6 IPsec Crypto Map Configuration

After configuring the global parameters of IPv6 IPsec, you can configure the specific IPv6 IPsec crypto map entry. A crypto map entry is a single policy that describes how certain traffic is to be secured. Each entry is given an index, which is used to sort the ordered list.

When a non-secured packet arrives on an interface, the crypto map associated with that interface is processed in order. If a crypto map entry matches the non-secured traffic, the traffic is discarded. When a secured packet arrives on an interface, its SPI is used to look up an SA. If an SA does not exist, or if the packet fails any of the security checks, it is discarded. If all checks pass, the packet is forwarded normally.

When a packet is to be transmitted on an interface, the crypto map set associated with that interface is processed in order. The first crypto map entry that matches the packet is used to secure the packet. If a suitable SA exists, that is used for transmission. Otherwise, an SA is established based on the manual key configuration.

Create the map and enter the map's configuration mode using the **ipv6 crypto map <name> <index> ipsec-manual** command. Once in the map's configuration mode, you can specify the IPv6 ACLs used to match traffic as well as the peer devices for use in tunnel mode, the manual key and SPI for IPsec processing, and the transform set to be used by the crypto map entry. Use the commands described in this section to configure the IPv6 IPsec crypto map entry.

1. Use the **match address ipv6 <ipv6 acl name>** command to associate an IPv6 ACL with the crypto map entry. The ACL designates the IPv6 traffic to be protected by this crypto map entry. When traffic sent or received at an interface is protected by this map, matching traffic is processed by IPsec. Traffic received that is unprotected, but matches the ACL, is dropped. Traffic that doesn't match the ACL is unaffected. You can only add a single ACL to a crypto map entry. By default, no IPv6 ACL is specified in the crypto map entry. If you enter this command with an ACL name that does not exist, an ACL is created that has an implicit permit, indicating it will match all traffic on the interface until it is configured further. ACLs must be IPv6 ACLs (for more information about configuring IPv6 ACLs, refer to [IPv6 ACLs on page 57](#)). Using the **no** form of this command removes the ACL from the crypto map entry and deletes all active SAs derived from this ACL in this map. Enter the command from the Crypto Map Entry Configuration mode as follows:

```
(config)#ipv6 crypto map VPN 10 ipsec-manual
(config-crypto6-map)#match address ipv6 ACLv61
(config-crypto6-map)#
```

2. Use the **set peer <ipv6 address>** command to define the peer device for an IPv6 crypto map entry when using IPsec tunnel mode. If no peer address is configured, the manual key crypto map entry is not complete. A peer IPv6 address is required for manual key crypto map entries. Specify the peer using the *<ipv6 address>* parameter. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. If the address is link-local, the link-local scope is assumed to be the interface having the crypto map assigned, and is the destination to which the protected

packet is routed. By default, no peer is assigned in a crypto map entry. Using the **no** form of this command removes the peer from the entry and deletes all active SAs established with this peer by this entry. To create a peer on the crypto map to use for IPsec tunneling, enter the command from the Crypto Map Entry Configuration mode as follows:

```
(config)#ipv6 crypto map VPN 10 ipsec-manual
(config-crypto6-map)#set peer 2001:DB8:1::1
(config-crypto6-map)#
```

- Next, assign the crypto map entry a transform set (defined by the **ipv6 crypto ipsec transform-set** *<name>* *<parameters>* command in the Global Configuration mode). Crypto map entries do not directly contain the transform configuration for securing data. Instead, the crypto map entry is associated with transform sets that contain specific security algorithms. Use the **set transform-set** *<name>* command to assign the previously created transform set to the crypto map entry. The *<name>* parameter is the name of the transform set to apply to the crypto map entry. For manual key crypto map entries, only one transform set can be specified. If no transform set is used in the crypto map entry, the entry is incomplete and will have no effect on the system. By default, no transform set is applied to the crypto map entry. Using the **no** form of this command removes the transform set from the crypto map entry configuration. To apply a transform set to the crypto map entry, enter the command as follows from the Crypto Map Entry Configuration mode:

```
(config)#ipv6 crypto map VPN 10 ipsec-manual
(config-crypto6-map)#set transform-set SET1
(config-crypto6-map)#
```

- Use the **set session-key [inbound | outbound] [ah <spi> <key> | esp <spi> [authenticator <key> | cipher <key>]]** to specify the inbound or outbound encryption and authentication keys for this crypto map entry. Specify whether the key will be used for inbound or outbound IPsec processing using the **inbound** or **outbound** parameters. Then specify the protocol and SPI using **ah <spi>** or **esp <spi>**. The *<spi>* parameter specifies the SPI for the IPsec SA to be created for the specified direction and protocol. A unique SPI is needed for both inbound and outbound traffic. The local system's inbound SPI and keys are the peer's outbound SPI and keys and the local system's outbound SPI and keys are the peer's inbound SPI and keys. Valid SPI range is **256 to 4294967295**. Next, specify the key using the *<key>* parameter. The type and number of keys required is defined by the transform set applied to the crypto map entry. ESP may require an authentication key (**authenticator <key>**), an encryption key (**cipher <keys>**), or both. The keys are specified in hexadecimal format without the leading **0x**. The key is not an ASCII string, but merely a string of bytes to be used in the algorithm specified by the transform set. [Table 11](#) outlines the key length requirements.



Each character, 0-9, a-f represents 4 bits, or half a byte. The number of characters in the key will be twice the key length in bytes.

Table 11. Session Key Length Requirements

Algorithm	Key Length
DES	64 bits in length; 8 hexadecimal bytes
3DES	192 bits in length; 24 hexadecimal bytes
AES-128-CBC	128 bits in length; 16 hexadecimal bytes

Table 11. Session Key Length Requirements (Continued)

Algorithm	Key Length
AES-192-CBC	192 bits in length; 24 hexadecimal bytes
AES-256-CBC	256 bits in length; 32 hexadecimal bytes
MD5	128 bits in length; 16 hexadecimal bytes
SHA1	160 bits in length; 20 hexadecimal bytes

By default, no session keys are defined for the crypto map entry. When configuring the session key, note that the inbound local SPI must equal the outbound remote SPI, and the outbound local SPI must equal the inbound remote SPI. Use the **no** form of this command to remove any defined encryption and authentication keys. Both the inbound and outbound keys must be set for the crypto map entry to be complete. The following example assumes a transform set using ESP with 3DES for encryption and SHA1 for authentication and uses this command in the Crypto Map Entry Configuration mode to define the inbound and outbound authentication and encryption keys:

```
(config)#ipv6 crypto map VPN 10 ipsec-manual
(config-crypto6-map)#set session-key inbound esp 300 cipher
32746524236738396A6E72286A21403472766E6668673565 authenticator
7235255E756768656D626B64686A333424782E3C
(config-crypto6-map)#set session-key outbound esp 400 cipher
3835363468676A656C7269676E2A2628676E622331246433 authenticator
696F37382A37676E65722334286D676E73642133
(config-crypto6-map)#
```

These commands complete the configuration of the crypto map entry. The next step in IPv6 IPsec configuration is to apply the crypto map to an interface.

IPv6 IPsec Interface Configuration

After configuring the global IPv6 IPsec parameters and the crypto map entry, the crypto map must be applied to an interface. Apply a crypto map to the interface using the **ipv6 crypto map <name>** command from the interface's configuration mode. The *<name>* parameter is the name of the map you want to apply to the interface. By default, no crypto maps exist on an interface. Using the **no** form of this command removes the map from the interface and all active SAs derived from the interface's crypto map are removed. You can only specify one crypto map per interface, and the crypto map is applied within the VRF instance to which the interface belongs. To apply the IPv6 IPsec crypto map, the interface must have IPv6 enabled. The interface must have an IPv6 address of appropriate scope to allow connectivity to the peer's address as specified in the crypto map's entries. To apply the crypto map to an interface, enter the command as follows from the interface's configuration mode:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6
(config-eth 0/1)#ipv6 crypto map MAP1
(config-eth 0/1)#
```

IPv6 IPsec Troubleshooting Commands

Various commands can be used from the Enable mode to view IPv6 IPsec configurations and create debug messages for IPv6 IPsec events. These commands are included in the following sections.

IPv6 IPsec Show Commands

Show commands can be used to view current IPv6 IPsec configurations, IPsec SAs, and the running configuration of IPv6 IPsec. The following sections describe the **show** commands available for IPv6 IPsec troubleshooting.

Use the **show ipv6 crypto [any-vrf | vrf <name>] ipsec sa [address <ipv6 address> | brief | map <name> | ospfv3 | [inbound | outbound]** command to display entries from the IPv6 IPsec SA database, which contains information about each IPsec SA. You can optionally specify that entries from all VRFs are displayed (**any-vrf** keyword) or that entries from a specific VRF are displayed (**vrf <name>**). If no VRF is specified, entries for the default VRF are displayed. You can optionally limit the output of this command to the SAs associated with the specified peer address using the **address <ipv6 address>** parameter. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. If no address is specified, entries for all peers are displayed. Use the optional **brief** keyword to display IPsec SAs in a condensed form (two lines per IPsec SA). You can optionally limit output of this command to display IPsec SAs created using a specific crypto map by using the **map <name>** parameter. If a map is not specified, SAs for all crypto maps are displayed. You can also optionally limit the output of this command to display only the SAs created for OSPFv3 using the **ospfv3** parameter. Lastly, you can optionally limit the output of this command to display SAs that protect either inbound or outbound traffic using the **inbound** and **outbound** parameters.

To display the IPsec SAs for all VRFs, enter the command from the Enable mode as follows:

```
>enable
```

```
#show ipv6 crypto any-vrf ipsec sa
```

```
2 current IPv6 IPsec SAs on default VRF
```

```
2 current IPv4 + IPv6 IPsec SAs on all VRFs (2 peak of 432 max)
```

```
IPv6 IPsec Security Associations:
```

```
Peer IPv6 Address: ::
```

```
  Crypto Map: VPN 10
```

```
  Direction: Inbound
```

```
  Encapsulation: ESP tunnel
```

```
  SPI: 0x000003E9 (1001)
```

```
  Internal ID: 1
```

```
  RX bytes: 740
```

```
Peer IPv6 Address: 2001:DB8:6723:2300::30
```

```
  Crypto Map: VPN 10
```

```
  Direction: Outbound
```

```
  Encapsulation: ESP tunnel
```

```
  SPI: 0x000003EA (1002)
```

```
  Internal ID: 2
```

```
  TX bytes: 740
```

To display IPsec SAs in a condensed form, enter the command from the Enable mode as follows:

```
>enable
```

```
#show ipv6 crypto ipsec sa brief
```

```
2 current IPv6 IPsec SAs on default VRF
```

```
2 current IPv4 + IPv6 IPsec SAs on all VRFs (2 peak of 432 max)
```

Bytes Processed	Peer IPv6 Address	Crypto Map	Remote ID
RX740	::	VPN 10	n/a
TX740	2001:DB8:6723:2300::30	VPN 10	n/a

Use the **show run ipv6 crypto [verbose]** command to display sections from the running configuration associated with IPv6 crypto functions. This includes any IPsec over IPv6 configuration. The optional **verbose** parameter displays more detailed information. To view IPv6 crypto running configuration, enter the command from Enable mode as follows:

```
>enable
```

```
#show run ipv6 crypto
```

```
Building configuration...
```

```
!
```

```
!
```

```
ipv6 crypto
```

```
!
```

```
ipv6 crypto transform-set T esp-3des esp-sha-hmac
  mode tunnel
```

```
!
```

```
ipv6 crypto map VPN 10 ipsec-manual
```

```
  match address ipv6 VPN6
```

```
  set peer 2001:DB8:6723:2300::30
```

```
  set transform-set T
```

```
  set session-key inbound esp 1001 cipher 1234567812345678123456781234567812345678123456781234567812345678
    authenticator 1234567890123456789012345678901234567890123456789012345678901234567890
```

```
  set session-key outbound esp 1002 cipher
```

```
    8765432187654321876543218765432187654321876543218765432187654321 authenticator
    09876543210987654321098765432109876543210987654321
```

```
!
```

```
ipv6 access-list extended VPN6
```

```
  permit IPV6 2001:DB8:6723:2100::/64 2001:DB8:6723:2200::/64
```

```
!
```

```
end
```

IPv6 IPsec Debug Commands

Debug commands are used to enable debug messages for IPv6 IPsec. Use the **debug ipv6 crypto [vrf <name>] ipsec** command to enable debug messages for IPv6 IPsec. You can optionally specify that debug messages are enabled only on a nondefault VRF instance using the **vrf <name>** parameter. If no VRF is specified, debug messages are enabled on the default VRF. To display debug event messaging for IPv6 IPsec on the default VRF, enter the command from Enable mode as follows:

```
>enable
#debug ipv6 crypto ipsec
```

IPv6 IPsec CLI Configuration Example

The following configuration example shows the relevant configuration necessary to configure an IPv6 over IPv6 IPsec tunnel. This example should be used for illustrative purposes only. You will need to make configuration changes to ensure the configuration will function in your network. The network topology for this example is shown below.

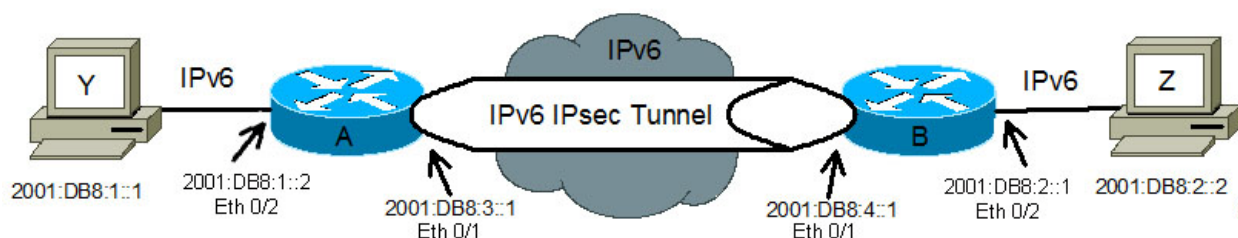


Figure 4. IPv6 over IPv6 IPsec in Tunnel Mode

The following is the sample configuration for both Router A and Router B:

Router A Configuration

```
!
ipv6 unicast-routing
ipv6 crypto
!
ipv6 crypto ipsec transform-set espaessha esp-aes-128-cbc esp-sha-hmac
!
ipv6 crypto map v6MAP 10 ipsec-manual
  set peer 2001:DB8:4::1
  set session-key inbound esp 10045 cipher 31323334353637383930313233343536 authenticator
  3132333435363738393031323334353637383930
  set session-key outbound esp 10046 cipher 31323334353637383930313233343536
  authenticator 3132333435363738393031323334353637383930
  set transform-set espaessha
  match address ipv6 v6protected
!
interface ethernet 0/2
  ipv6 address 2001:DB8:1:2/64
!
interface ethernet 0/1
  ipv6 address 2001:DB8:3:1/64
  ipv6 crypto map v6MAP
!
```

```
ipv6 access-list extended v6protected
  permit ipv6 2001:DB8:1::/64 2001:DB8:2::/64
!
ipv6 route 2001:DB8:2::/64 2001:DB8:3::2
!
```

Router B Configuration

```
!
ipv6 unicast-routing
ipv6 crypto
!
ipv6 crypto ipsec transform-set espaessha esp-aes-128-cbc esp-sha-hmac
!
ipv6 crypto map v6MAP 10 ipsec-manual
  set peer 2001:DB8:3::1
  set session-key inbound esp 10046 cipher 31323334353637383930313233343536 authenticator
  3132333435363738393031323334353637383930
  set session-key outbound esp 10045 cipher 31323334353637383930313233343536
  authenticator 3132333435363738393031323334353637383930
  set transform-set espaessha
  match address ipv6 v6protected
!
interface ethernet 0/2
  ipv6 address 2001:DB8:2:1/64
!
interface ethernet 0/1
  ipv6 address 2001:DB8:4:1/64
  ipv6 crypto map v6MAP
!
ipv6 access-list extended v6protected
  permit ipv6 2001:DB8:2::/64 2001:DB8:1::/64
!
ipv6 route 2001:DB8:1::/64 2001:DB8:4::2
!
```


IPv6 General Utility Commands

The general utility commands for IPv6 used in AOS include various **clear**, **show**, and **debug** commands, as well as **ping** and **traceroute** commands. These commands are described in the following sections. For utility or troubleshooting commands that relate to a specific IPv6 feature, refer to the appropriate section in this document.

IPv6 Clear Commands

The **clear** commands associated with IPv6 in AOS include clearing IPv6 neighbor cache entries, prefix information, IPv6 route cache entries, and many others. All clear commands are entered from the Enable mode prompt. The IPv6 **clear** commands are described in the following section.

1. Use the **clear ipv6 neighbors** [*<interface>* | **vrf** *<name>*] [*<ipv6 address>*] [**statistics**] command to clear dynamic entries from the neighbor cache. Specify whether you are clearing entries for a specific VRF or a specific interface using either the **vrf** *<name>* or *<interface>* parameters. Specify interfaces in the *<interface>* *<slot/port | interface id>* format. You can optionally specify whether all entries are cleared or if entries for a specific IPv6 address are cleared by specifying an IPv6 address. If no address is specified, all entries are cleared. If no VRF is specified, entries on the default unnamed VRF are cleared. If no interface is specified, all entries for all interfaces on the VRF are cleared. Using the **statistics** parameter specifies that statistics for the ND cache and protocol interaction are cleared. By default, if no options are specified, entering this command clears all neighbor cache entries on all interfaces assigned to the default VRF. To clear all neighbor cache entries, enter the command as follows:

```
#clear ipv6 neighbors
```

2. Use the **clear ipv6 interface** *<interface>* **prefix** command to clear IPv6 address prefix information from a specified interface. Specify interfaces in the *<interface>* *<slot/port | interface id>* format. For example, to clear all IPv6 address prefix information from an Ethernet interface, enter the command as follows:

```
#clear ipv6 interface ethernet 0/1 prefix
```

3. Use the **clear ipv6 routers** [**conflict**] [**vrf** *<name>*] [*<interface>*] to clear the list of routers learned from RA messages from locally reachable routers. By default, this command clears all learned routers from all interfaces on the default unnamed VRF when no options are specified. You can optionally use this command to clear the learned router list from all interfaces or a named interface on the default VRF (by specifying an interface or using no options), to clear RAs from all interfaces or a specified interface on a specified VRF (by specifying a VRF and optionally specifying an interface), or to clear learned routers from all interfaces or a single interface (by specifying a VRF and specifying an interface). Interfaces are specified in the *<interface>* *<slot/port | interface id>* format. For example, to clear learned routers from all interfaces on the default VRF, enter the command as follows:

```
#clear ipv6 routers
```

Use the optional **conflict** parameter to clear the list of learned routers with misconfigurations from locally reachable routers. Enter the command as follows:

```
#clear ipv6 routers conflict
```

- Use the **clear ipv6 cache [vrf <name>] [counters]** command to clear all the route cache entries in a given VRF. If the optional **counters** parameter is specified, then only the use-count statistics are cleared on each route cache entry. The optional **vrf <name>** parameter specifies a nondefault VRF on which to clear all the route cache entries. If the VRF is not specified, all entries on the default VRF are cleared. Enter the command as follows to clear all IPv6 route cache entries:

```
#clear ipv6 cache
```

- Use the **clear ipv6 mld traffic [vrf <name>]** command to reset the IPv6 MLD traffic counters to zero. The optional **vrf <name>** parameter specifies a nondefault VRF on which to reset the MLD traffic counters. If the VRF is not specified, all counters on the default VRF are reset. Enter the command as follows to reset the IPv6 MLD traffic counters:

```
#clear ipv6 mld traffic
```

IPv6 Show Commands

The **show** commands associated with IPv6 in AOS include displaying information for IPv6 interfaces, route tables, IP statistics, the running configuration, the IPv6 route cache, and many others. All show commands are entered from the Enable mode prompt. The IPv6 **show** commands are described below.

- Use the **show ipv6 interface [brief | <interface> | <interface> prefix]** command to display status information for IPv6 interfaces. This information includes IPv6 addressing, configured parameters, and any IPv6 capabilities in use. The **brief** keyword specifies that the output is abbreviated. If this keyword is not used, all IPv6 information (except prefix details) is displayed. The **<interface>** parameter limits the output to a single interface. Interfaces are specified in the **<interface> <slot/port | interface id>** format. The **prefix** keyword specifies that the list of prefixes for a specified interface are displayed. For example, to display IPv6 status information for an Ethernet interface, enter the command as follows:

```
#show ipv6 interface brief ethernet 0/1
```

```
eth 0/1 [UP/UP]
  FE80::2AO:C8FF:FE61:3082
  2003::2AO:C8FF:FE61:3082
```

- Use the **show ipv6 route [vrf <name> | <ipv6 address> | <ipv6 prefix/prefix-length> | static | connected]** command to display the contents of the IPv6 route table. This table contains information about IPv6 networks and how to reach them. The output of this command can be limited by optionally specifying a VRF for which to display information (otherwise information for the default unnamed VRF is displayed), by optionally specifying an IPv6 address, and IPv6 prefix, using the **static** keyword to only display static routes, or by using the **connected** keyword to only display connected routes. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The prefix value is specified in colon hexadecimal format (**X:X:X/<Z>**), for example, **2001:DB8:3F::/48**. The prefix length (**<Z>**) is an integer with a value between **0** and **128**. For example, to display the IPv6 route table for IPv6 address prefix **2001::/64**, enter the command as follows:

```
#show ipv6 route 2001::/64
```

```
Routing entry for 2001::/64
  Known via "static"
  Distance 1, metric 0
  Routing Next Hop(s):
    2002::1, via eth 0/1
  Route metric is 0
```

3. Use the **show ipv6 route [vrf <name>] summary** command to display a summary of the IPv6 route table statistics. You can display the statistics for a specific VRF using the **vrf <name>** parameter, or if no VRF is specified, statistics for the default unnamed VRF are displayed. To display a summary of IPv6 route table statistics, enter the command as follows:

#show ipv6 route summary

Route Source	FIB	Local-RIB
Connected	3	3
Other	18	18
Total	21	21

4. Use the **show ipv6 traffic** command to display the system-wide statistics for the IPv6 and ICMPv6 protocols. Enter the command as follows:

#show ipv6 traffic

IPv6 statistics:

```
Rcvd:      0 total, 9 local destination
           0 header errors, 0 address errors
           0 unknown protocol, 0 discards
           0 truncated, 0 bad hop counts
Sent:      0 locally generated, 59 forwarded
           0 no route, 0 discards
Frag:      0 reassemble required, 0 reassembled, 0 couldn't reassemble
           0 created, 0 fragmented, 0 couldn't fragment
```

5. Use the **show ipv6 mld traffic [vrf <name>]** command to display the IPv6 MLD traffic counters. The optional **vrf <name>** parameter specifies a nondefault VRF for which to display the MLD traffic counters. If a VRF is not specified, MLD traffic counters for the default VRF are displayed. To display MLD traffic counters on the default VRF, enter the command as follows:

#show ipv6 mld traffic

MLD Traffic Counters

Elapsed time since counters cleared: Never cleared

	Sent	Received
Valid MLD Packets	3263	1628
Queries	0	1628
Reports	3263	0
Leaves	0	0

Errors:

Malformed Packets	0
Non link-local source	0
Hop limit not equal to 1	0

6. Use the **show ipv6 mld groups link-local [vrf <name>] [<interface>]** command to display the known multicast groups used by IPv6 features on the AOS router. The optional **vrf <name>** parameter specifies a nondefault VRF for which to display the MLD groups. If a VRF is not specified, MLD groups for the default VRF are displayed. The optional **<interface>** parameter specifies that only the MLD groups registered on the interface are displayed. If no interface is specified, all MLD groups are

displayed. Interfaces are specified in the `<interface> <slot/port | interface id>` format. To display the MLD groups used on the Gigabit Ethernet interface **0/2.1**, enter the command as follows:

#show ipv6 mld groups link-local giga-eth 0/2.1

MLD Connected Group Membership

Group Address	Interface	Uptime	Expires
FF02::1	giga-eth 0/2.1	13h32m32s	Never
FF02::2	giga-eth 0/2.1	13h32m32s	Never
FF02::1:FF00:1234	giga-eth 0/2.1	13h32m32s	Never
FF02::1:FF01:2CC	giga-eth 0/2.1	13h32m32s	Never

- Use the **show ipv6 cache [vrf <name>]** command to display the contents of the route cache for each interface in a given VRF. The route cache contains information about which egress interface, IPv6 gateway address, and MAC address to use when forwarding packets to a given destination. The optional **vrf <name>** parameter specifies a nondefault VRF for which to display route cache information. If a VRF is not specified, route cache information is displayed for the default VRF. To display route cache statistics, enter the command as follows:

#show ipv6 cache

INGRESS: giga-eth 0/1.1001

DEST: 2001:1111:2222:3333:4444:5555:6666:7777, EGRESS: giga-eth 0/1.1002,
COUNT: 1000000000, GATEWAY: 2001:1111:2222:3333:4444:5555:6666:0001,
MAC: 00:a0:c8:00:00:01, ID: 0x01000000

DEST: 2001::1, EGRESS: ppp 1,
COUNT: 100, GATEWAY: 2001::1:0001,
MAC: n/a, ID: 0x01010000

DEST: 2001::2, EGRESS: ppp 1,
COUNT: 100, GATEWAY: 2001::1:0001,
MAC: n/a, ID: 0x01020000

IPv6 Debug Commands

The **debug** commands associated with IPv6 in AOS can be useful in troubleshooting IPv6 configurations. These commands can enable debug messaging for sent and received IPv6 packets, IPv6 routing table events, and much more. All debug commands are entered from the Enable mode prompt. The IPv6 **debug** commands are described below.



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

- Use the **debug ipv6 routing [vrf <name>]** command to enable debug messages for IPv6 routing table events. To enable debug messages for an IPv6 firewall on a named VRF, specify the VRF name with the **vrf <name>** parameter. If no VRF is specified, IPv6 routing table events for the default VRF are displayed. Enter the command as follows:

#debug ipv6 routing

- Use the **debug ipv6 mld [packet | events] [interface <interface>]** command to display information related to MLD queries and their generated responses. The **packet** parameter specifies that decoded packet information is displayed. This option allows you to view the contents of the MLD messages as

well as the groups being queried or reported. The **events** parameter specifies that events related to MLD activity (such as timer starts or compatibility mode changes) are displayed. The optional **interface** *<interface>* parameter specifies that displayed output is limited to the interface. Interfaces are specified in the *<interface> <slot/port | interface id>* format. Enter the command as follows for information about MLD packets:

#debug ipv6 mld packet

```
2014.04.20 17:58:05 MLD.PKT giga-eth 0/2.1 Receive MLD packet, len=28
type=130 (query), code=0, cksum=0x231d
maxDelayMs=5000, mcast=: (general)
sFlag=0, QRV=2, QQIC=30, numSources=0
2014.04.20 17:58:05 MLD.PKT giga-eth 0/2.1 Transmit MLD packet, len=88
type=143 (v2 report), code=0, cksum=0x8f49
numRecords=3
recordType=2, auxDataLen=0, numSources=0, mcast=FF02::2
recordType=2, auxDataLen=0, numSources=0, mcast=FF02::1:FF01:2CC
recordType=2, auxDataLen=0, numSources=0, mcast=FF05::2
```

Enter the command as follows for information about MLD events:

#debug ipv6 mld events

```
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF01::1
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Suppressed advertisement for FF01::1
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF02::1
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Suppressed advertisement for FF02::1
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF01::2
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Suppressed advertisement for FF01::2
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF02::2
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add FF02::2 to pending join queue
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Delayed report timer started; robust=1,
interval=491/1000
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Delayed report timer started; robust=2, interval=91/1000
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF05::2
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add FF05::2 to pending join queue
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add multicast group: FF02::1:FF01:2CC
2014.04.20 06:13:45 MLD.EVT giga-eth 0/2.1 Add FF02::1:FF01:2CC to pending join queue
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF02::1:FF01:2CC
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Delayed report timer started; robust=1,
interval=356/1000
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Delayed report timer started; robust=2,
interval=607/1000
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Add FF02::1:FF01:2CC to pending leave queue
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF01::1
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF02::1
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF01::2
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF02::2
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Add FF02::2 to pending leave queue
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Remove multicast group: FF05::2
2014.04.20 06:14:36 MLD.EVT giga-eth 0/2.1 Add FF05::2 to pending leave queue
```

IPv6 Ping and Traceroute Commands

In IPv6, AOS has added the capability to use the IPv6 **ping** and **traceroute** features. Each feature has added an **ipv6** keyword to the command syntax, as well as a few other IPv6 changes. These changes (although not the entire **ping** or **traceroute** capability) are described in the following section.

The IPv6 **ping** command uses the following syntax: **ping ipv6** *<ipv6 address>* [*<interface>*] [**hop-by-hop-option**] [**destination-option**]. This command can be entered from either the Basic mode prompt or the Enable mode prompt. The *<ipv6 address>* parameter specifies the destination you are attempting to ping. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The optional *<interface>* parameter specifies the link-local interface for the ping request. If you are using a link-local IPv6 address, you must specify the interface. Interfaces are specified in the *<interface>* *<slot/port | interface id>* format. The optional **hop-by-hop-option** parameter specifies that the hop-by-hop option in the ICMPv6 echo request packet is included. Using this option typically causes intermediate routers to process switch the packets, potentially detecting switching issues in those devices. The optional **destination-option** parameter specifies that the destination option in the ICMPv6 echo request packet is included. To use the IPv6 **ping** command, enter the command as follows:

```
#ping ipv6 2001:DB8:1::1
```

The IPv6 **traceroute** command uses the following syntax: **traceroute ipv6** *<ipv6 address>* [*<interface>*]. This command can be entered from either the Basic Mode prompt or the Enable mode prompt. The *<ipv6 address>* parameter specifies the IPv6 address of the target whose path is traced. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The optional *<interface>* parameter specifies the link-local interface for the traceroute request. If you are using a link-local IPv6 address, you must specify the interface. Interfaces are specified in the *<interface>* *<slot/port | interface id>* format.

To use the IPv6 **traceroute** command, enter the command as follows:

```
#traceroute ipv6 2001:DB8:1::1
```

IPv6 Neighbor Discovery (ND) in AOS

ND is a set of messages and processes that determine relationships between neighboring nodes. In IPv6, ND replaces ARP, ICMPv4 Router Discovery, and ICMPv4 Redirect. ND is used by nodes for address resolution, determining link-layer address changes, and determining neighbor reachability. ND is used by hosts to discover neighboring routers and to automatically configure addresses, address prefixes, and other configuration parameters. ND is used by routers to advertise their presence, configuration parameters, and on-link prefixes in router advertisement (RA) messages, and to inform hosts of better next-hop addresses to forward packets for a specific destination (a redirect).

ND is composed of many parts. In the host, there are typically two caches and two lists associated with ND. Each host has access to a Neighbor cache, which stores the on-link IPv6 address of a neighbor, its corresponding link-layer address, and an indication of the neighbor's reachability state. The neighbor cache is the equivalent to the ARP cache in IPv4. In addition, each host has access to a destination cache (which is often integrated into the IPv6 route table) that stores information on the next-hop IPv6 addresses for destinations to which traffic has recently been sent. Each host also maintains a prefix list, which contains the on-link prefixes, and a default router list, that lists addresses corresponding to on-link routers that advertise themselves as default routers.

ND uses ICMPv6 messaging structure and ICMPv6 message types **133** through **137**. All ND messages are sent with a hop limit of **255** to prove that the messages originated on-link. Receivers will not process ND messages if they are received at a hop limit less than 255 because that indicates the message was sent through a router. ND operates by using router discovery to find the prefix and parameters of nearby routers, as well as to provide IPv6 address autoconfiguration, and then ND performs address resolution, next-hop determination, neighbor unreachability detection (NUD), duplicate address detection (DAD), and redirect functions. These operations are all performed by the sending and receiving of ICMPv6 messages. There are five message types associated with ND: router solicitation (RS) messages, router advertisement (RA) messages, neighbor solicitation (NS) messages, neighbor advertisements (NA) messages, and redirect messages.

Router Solicitation (RS) ND Messages

RS messages are sent by hosts on the local link network. Hosts use these messages to inquire about the presence of a router on the link. In addition, RS messages can be used when hosts are booting up to request RA messages that will contain autoconfiguration information for the host interfaces. These messages are sent to an all routers multicast address (FF02::2). In these messages, the source IPv6 address is either a link-local address or an unspecified IPv6 address (::). RS messages are identified by a value of **133** in the Type field of an ICMPv6 packet header.

Router Advertisement (RA) ND Messages

RA messages are sent in response to RS messages and have a value of **134** in the Type field of the ICMPv6 packet header. These messages are sent periodically on configured interfaces on IPv6 routers. RA messages include IPv6 prefixes that nodes on the local link can use for IPv6 address autoconfiguration, the lifetime of each IPv6 prefix, the flags used to indicate that stateful or stateless autoconfiguration can be implemented, the default router information, and additional information, such as hop limit and MTU settings that hosts should use. Stateless Address Autoconfiguration (SLAAC) is based on the RA advertised prefixes with a 64-bit prefix length. The 64-bit prefix is combined with the Interface ID to form

the IPv6 address assigned to the receiving interface. Information in RA messages can be manually configured, and the configurable parameters include the time interval between RA messages, the default router lifetime, the network prefixes used on the link, and the amount of time a node is considered reachable. Each of these parameters are configured for RA messages on a particular interface.

Neighbor Solicitation (NS) ND Messages

NS messages are sent on the local link and are used to discover the link-layer address of nodes on the same local link as the sending node. NS messages are also used in duplicate address detection (DAD). The source address in the NS message is the IPv6 address of the sending node. The destination address is the solicited-node multicast address that corresponds to the IPv6 address of the destination node. These messages are used to determine neighbor reachability and work with NUD. These messages can also include the link-layer address of the sending node. NS messages have a value of **135** in the Type field of the ICMPv6 packet header.

Neighbor Advertisement (NA) ND Messages

NA messages are sent in response to NS messages, and these messages include the link-layer address of the node sending the NA message. NA messages are also used in DAD. When NA messages are received by the node that sent the NS message, the two nodes can begin to communicate. NA messages also indicate the reachability of a neighbor, and can be used when the link-layer IPv6 address of a node has changed. NA messages have a value of **136** in the Type field of the ICMPv6 packet header.

Redirect Messages

ND redirect messages function similarly to redirect messages in ICMPv4. These messages have a value of **137** in the Type field of the ICMPv6 packet header. These messages are sent when better first-hop nodes on the destination path are discovered.

Neighbor Unreachability Detection (NUD) in ND

NUD detects whether a neighbor (destination or first hop to the destination) is reachable. Reachability is determined by the receipt of a NA messages with the solicited flag set while the node is waiting for a response from a NS message. NUD begins when a packet needs to be sent and the neighbor to which the packet is being sent is in a STALE state. Neighbors enter the STALE state when their reachable time expires. The reachable time for neighbors is conveyed to hosts in the RA messages. NUD uses the same exchange as ARP in IPv4, except NS and NA messages are sent to the unicast address listed in the neighbor cache.

Duplicate Address Detection (DAD) in ND

DAD is used by devices to determine if IPv6 addresses are unique before they are applied to interfaces. DAD uses NS and NA messages, and is able to detect duplicate unicast addresses. The Target Address fields in the NS messages are set to the IPv6 address for which duplication is being detected. Destination addresses for DAD in NS messages are the solicited-node multicast version of the address being tested. Source addresses for DAD are set to the IPv6 unspecified address (::). If no NA message is received that corresponds to the address being tested, then the IPv6 address is considered to be unique and it can be applied to the IPv6 interface on the node.

Table 12 describes the neighbor detection functions in IPv4 and their equivalents in IPv6.

Table 12. Neighbor Detection Functions in IPv4 and IPv6

IPv4 Neighbor Detection	IPv6 Neighbor Detection
ARP Request Message	Neighbor Solicitation (NS) Message
ARP Reply Message	Neighbor Advertisement (NA) Message
ARP Cache	Neighbor Cache
Gratuitous ARP	Duplicate Address Detection (DAD)
Router Solicitation Messages Are Optional	Router Solicitation (RS) Is Required
Router Advertisement Messages Are Optional	Router Advertisements (RA) Are Required
Redirect Messages	Redirect Messages

Configuring IPv6 ND in AOS

The following commands are used to configure the IPv6 ND protocol and its operation on specific interfaces.

1. Specify the number of NS messages that are sent by the interface when performing DAD using the **ipv6 nd dad attempts** *<number>* command. This command is effectual when the interface is in either router or host mode. The *<number>* parameter is the number of NS messages. Valid range is **0** to **10**. By default, **1** NS message is sent. A value of **0** disables DAD on the interface. Using the **no** form of this command returns to the default value. DAD in AOS is performed when an interface transitions state from DOWN to UP or when manually configuring an address. When performing DAD because of an interface transition, DAD will happen immediately after the interface transition and again 40 seconds later to cooperate with the port being connected to an Ethernet switch. To specify the number of NS messages sent by the interface while performing DAD, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 nd dad attempts 3
```

- Specify the M flag in RAs using the **ipv6 nd managed-config-flag** command. This command is only effectual when the interface is in router mode. The M flag instructs hosts receiving the RA that they can use stateful DHCPv6 to configure addresses and non-address information. The **no** form of this command disables the setting of the M flag. By default, the M flag is not set in RAs. To set the M flag, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 nd managed-config-flag
```



If you specify that the M flag is set in RAs, you do not need to set the O flag (it becomes redundant).

- Specify the interval between transmissions of certain ND messages and control what ND value is advertised in RAs using the **ipv6 nd ns-interval <value>** command. This command is effectual whether the interface is host or router mode. This command controls the spacing of NS messages for functions, such as address resolution, reachability detection, and DAD. For DAD, it also serves as the amount of time after the last transmission before the detection phase of autoconfiguration terminates. In addition, the command controls the interval between unsolicited NA messages. The *<value>* parameter specifies the time (in milliseconds) between neighbor message transmissions. Valid range is **1000** to **3600000** ms. By default, the interval is set to **1000** ms for internal use and **0** (unspecified) is sent in RA messages. Using the **no** form of this command returns the interval to the default value. To change the interval between RA messages, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 nd ns-interval 2000
```

- Specify the O flag value in RA messages using the **ipv6 nd other-config-flag** command. This command is only effectual when the interface is in router mode. When the O flag is set, hosts receiving the RA messages are instructed that they may use stateless DHCPv6 to receive information that is not IPv6 addressing information, and to use some other method (whether through manual configuration, SLAAC, etc.) for addressing information. By default, the O flag is not set in RA messages. Using the **no** form of this command disables the O flag setting. To set the O flag in RA messages, enter the command from the interface's configuration mode as follows:

```
(config)#interface ethernet 0/1
(config-eth 0/1)#ipv6 nd other-config-flag
```



If the M flag is set for RA messages, you do not need to set the O flag (it becomes redundant).

- Specify or modify the IPv6 prefixes used in RA messages using the **ipv6 nd prefix [<ipv6 prefix/prefix-length> | default] [<valid lifetime> | infinite] [<preferred lifetime> | infinite] [off-link] [no-rtr-address] [no-autoconfig] [no-advertise] [no-onlink]** command. This command works for both routers and hosts, but in host implementations it is used to manually add on-link prefixes that do not have an address or to make off-link a prefix generated by an address command. Hosts do not send RA messages, so the command only adds prefixes to RA messages when the interface is in router mode.

This command can also be used to change the defaults used on configured prefixes when all options are not specified.



*Changing the prefix defaults will affect prefixes derived from configured IPv6 addresses, as well as prefixes configured using the **ipv6 nd prefix** command.*

Prefixes for IPv6 addresses configured on a router interface are automatically eligible to be advertised on that interface using system or configured default values without having to enter a prefix command. To impose additional controls on those prefixes, an entry must be made using this command with the desired settings.

The *<ipv6 prefix/prefix-length>* parameter specifies the prefix and length to be advertised. 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 **default** parameter is used to change the default settings for the IPv6 prefix parameters. Changing these settings can be useful when multiple prefixes are implemented that will use the same set of parameters. When configuring IPv6 prefixes, the prefix default values are only used if no parameters are specified after the IPv6 prefix and length are specified (for example, **ipv6 nd prefix 2001:DB8::/64**). If additional parameters are specified, any unspecified parameters use the system default values rather than the configured default values. When the default values are changed, any prefix that uses them will also change. Using this command to change prefix default values also affects prefixes derived from configured IPv6 addresses on the interface.

The optional *<valid lifetime>* parameter specifies the valid lifetime to advertise for this route in each advertisement. Hosts will reset the lifetime back to this value each time the route is advertised, and they will keep this prefix until the valid lifetime expires. You can set the valid lifetime value with the *<valid lifetime>* parameter by entering a time value between **0** and **4294967295** seconds, or you can specify that the prefix does not age by entering the **infinite** keyword. By default, prefixes have a valid lifetime of **2592000** seconds.

The optional *<preferred lifetime>* parameter specifies the preferred lifetime to advertise for this route in each advertisement. Hosts will reset the lifetime back to this value each time the route is advertised, and they will keep the prefix in the preferred state during this time period. After the preferred time period expires, the prefix transitions to the deprecated state where it remains until the valid lifetime expires and the route is removed. The preferred lifetime value must be set to be shorter than the valid lifetime value. You can set the preferred lifetime with the *<preferred lifetime>* parameter by entering a time value between **0** and **4294967295** seconds, or you can specify that the prefix does not age by entering the **infinite** keyword. By default, prefixes have a preferred lifetime of **604800** seconds.

The optional **off-link** parameter sets the L flag (on-link flag) value to **0** in RA messages. When the L flag is set to 0, the advertisement makes no statement about on-link or off-link properties of the prefix. When the L flag is set, the prefix is considered on-link and locally reachable by hosts on the link (meaning a router is not needed). Hosts attached to the link will add on-link prefixes to their prefix list or route table. When off-link is not specified, a connected route is added to the route table of this router for this prefix. When off-link is specified, no route is added to the route table. By default, prefixes are advertised as on-link with the L flag set.

The optional **no-rtr-address** parameter sets the R flag (router flag) of the RA to **0** and does not include the full router address in the advertisement. The router address is typically included in the RA to assist in mobile IP environments. By default, the R flag is set and the router address is sent in RA messages.

The optional **no-autoconfig** parameter sets the A flag of the RA to **0**, indicating that hosts may not create an address for this prefix using SLAAC. If the A flag is set to **1** (the default setting), hosts perform SLAAC to generate an address based on the prefix. This parameter only affects hosts receiving the RA, it does not affect the operation of the local router.

The optional **no-advertise** parameter specifies that the prefix is excluded from RA messages. By default, the prefix is included in RA messages. The **no-onlink** parameter informs the router that the prefix is not to be used for on-link determination.

By default, all prefixes derived from the interface's configured IPv6 addresses are advertised using the system default values. Using the **no** form of this command removes the specified prefix configuration from the interface. To change the default values and behaviors of the prefixes included in RA messages, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd prefix default infinite infinite off-link no-autoconfig
```

- Specify the interval between the transmission of RA messages using the **ipv6 nd ra interval** [**<max time>** [**<min time>**] | **msec** **<max time>** [**<min time>**]] command. This command is only effectual when the interface is in router mode. When specifying the interval, you must specify the maximum interval, and you can optionally specify the minimum interval. In addition, you can choose to set these time values in either seconds or ms. To specify the time in seconds, use the **<max time>** and optional **<min time>** parameters without the **msec** keyword. The **<max time>** range is **4** to **1800** seconds, and the **<min time>** range is **3** seconds to 75 percent of the configured maximum time value. By default, the interval is set in seconds and has a maximum interval time of **200** seconds and a minimum interval time of 75 percent of the maximum seconds value, but not less than **3** seconds. To set the interval between RA messages in ms, enter the **msec** keyword followed by the **<max time>** and optional **<min time>** parameters. Using the ms value can be useful in mobile IP environments. The valid range for **msec** **<max time>** is **70** to **1800000** ms, with a default value of none. The valid range for **msec** **<min time>** is **30** ms to 75 percent of the configured ms maximum interval, with a default value of 75 percent of the maximum ms value but not less than **30** ms. To specify the interval between RA messages (in the following example the interval is set to **300** seconds maximum), enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd ra interval 300
```



*If this router is used as a default router, the interval between RA messages should not be set to a larger value than the RA lifetime set by the **ipv6 nd ra lifetime** command, which has a default value of **1800** seconds.*

- Specify that the Advertisement Interval Option is sent from the router using the **ipv6 nd advertisement-interval** command. This command is only effectual when the interface is in router mode. Sending of the Advertisement Interval Option should be enabled when the router is functioning in a mobile IP environment to aid movement detection by mobile nodes. This option contains the current value of the maximum router advertisement interval as set by the **ip nd ra interval** command. By default, Advertisement Interval Options are not sent in RA messages. Using the **no** form of this command returns to the default setting. To enable this feature, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd advertisement-interval
```

- Specify the router lifetime value in RA messages using the **ipv6 nd ra lifetime [*<value>* | default-route]** command. This command is effectual when the interface is in router mode. A non-zero value indicates to other nodes that this router can be used as a default router. A value of **0** indicates this router is not a default router. A non-zero value should be larger than the router advertisement interval specified in the **ipv6 nd ra interval** command. Valid value range is **0** to **9000** seconds. The **default-route** parameter specifies the RA lifetime is **0** if no default route exists on any IPv6 interface. Using the **no** form of this command returns the value to the default setting (**1800** seconds). To change the router lifetime value in RA messages, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd ra lifetime 3000
```

- Specify whether router advertisements will be suppressed using the **ipv6 nd ra suppress** command. This command is only effectual for interfaces in router mode. Using this command disables the sending of RA messages through the interface. Using the **no** form of this command enables the sending of RA messages. By default, RA messages are not suppressed. When IPv6 routing is not enabled on the router, or when implemented in a host-only mode, the default setting is to suppress advertisements on all interface types. To suppress RA messages on the interface, enter the command from the interface configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd ra suppress
```

- Specify the value advertised for Reachable Time in RA messages, and also set the Base Reachable Time used internally by the router using the **ipv6 nd ra reachable-time *<value>*** command. This command is effectual for interfaces in either router or host mode. For hosts, this value sets the internal reachable time used by the host if no RAs are received specifying a different value. For routers, the value indicates the amount of time a device is considered reachable after having received a reachability confirmation in NUD. The time value is specified in milliseconds, with a valid range of **0** to **3600000** ms. By default, a value of **0** is advertised by the router, indicating the reachable time is unspecified in RAs, and a value of **30000** ms is used locally by the node. Using the **no** form of this command returns the reachability value to the default setting. To change the advertised reachability time, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd ra reachable-time 50000
```

- Specify the default router preference value sent in RA messages using the **ipv6 nd router-preference [high | medium | low]** command. This command helps receivers of the RA messages to determine the preference of one router over another as a default router in environments with multiple routers. The **high** parameter specifies the preference value as high, **medium** specifies the preference as medium, and **low**

specifies the preference as low. By default, the RA messages give the interface a **medium** preference value. Using the **no** form of this command returns the preference value to the default setting. To change the advertised default router preference for the interface, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd router-preference high
```

12. Specify the maximum amount of time an unused (STALE) neighbor entry remains in the neighbor cache using the **ipv6 nd purge-timer** *<value>* command. This command is effectual for interfaces in either router or host mode. A neighbor entry is typically purged when NUD is invoked and the neighbor is determined to no longer be reachable. However, NUD is not performed on idle (STALE) neighbor entries, so this command provides a method for purging unused entries after a specified amount of time. This time value is specified in minutes, with a valid range of **10** to **1440** minutes. By default, idle (STALE) entries are purged after **1440** minutes (24 hours). Using the **no** form of this command returns the purging interval to the default value. To change how often idle entries are removed from the neighbor cache, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd purge-timer 800
```

13. Specify the maximum number of incomplete neighbor entries stored in the neighbor cache using the **ipv6 nd cache maximum-incomplete** *<number>* command. This command is effectual for interfaces in either router or host mode. By default, the incomplete ND entries can take at a maximum one-third of your possible ND cache entries (varies by product). Valid range is **1** to **321** entries. Using the **no** form of this command returns the maximum number of stored incomplete entries to the default value. To change the maximum number of incomplete neighbor entries stored in the neighbor cache, enter the command from the interface's configuration mode as follows:

```
(config)#interface eth 0/1
(config-eth 0/1)#ipv6 nd cache maximum-incomplete 150
```

Troubleshooting ND in IPv6

The following **show** and **debug** commands can be used to view ND statistics and to verify ND configuration on the AOS unit. These commands are executed from the Enable mode.

1. Use the **show ipv6 neighbors** [**vrf** *<name>* | *<interface>*] [*<ipv6 address>*] [**statistics**] command to display the IPv6 ND cache. This cache contains information about IPv6 nodes that have been added to the neighbor cache, including the link-layer IPv6 address and the reachability state of that neighbor. Neighbor entries are created when there is a packet to send to an on-link destination, or when an NS, RS, or RA message is received. You can limit this output by specifying a nondefault VRF, an interface, or an IPv6 address. If none of these options is specified, then information for neighbors on all interfaces on the default unnamed VRF are displayed. Interfaces are specified in the *<interface>* *<slot/port | interface id>* format. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. In addition, you can specify that cache statistics and protocol interaction information are displayed by using the optional **statistics** parameter.

For example, to display the ND cache, enter the command as follows:

#show ipv6 neighbors

IPv6 Address	Age	Link-layer Addr	State	Interface
2002::1	0	000f.352.3.2aba	REACH	eth 0/0
2003::ED9A:D1A3:BB9B:BDFF	0	0013.ce61.65b9	REACH	eth 0/1
20FF:11::ED9A:D1A3:BB9B:BDFF	18	0013.ce.61.65b9	STALE	eth 0/1
FE80::213:CEFF:FE61:65B9	10	0013.ce.61.65b9	DELAY	eth 0/1
FE80::20F:35FF:FE2E:2ABA	1	000f.352e.2aba	DELAY	eth 0/1

- Use the **show ipv6 routers [vrf <name> | conflict | <interface>]** command to display information learned from RA messages received from locally reachable routers. You can optionally specify that information for a certain VRF is displayed, using the **vrf <name>** parameter. If no VRF is specified, information for the default VRF is displayed. You can also optionally specify that information for a certain interface is displayed using the **<interface>** parameter. Interfaces are specified in the **<interface> <slot/port | interface id>** format. The optional **conflict** parameter specifies that only information about routers whose advertisements are in conflict with the interface settings are displayed. To display the router information for all interfaces on the default unnamed VRF, enter the command as follows:

#show ipv6 routers

```
Router FE80::20F:35FF:FE2E:2ABA on Ethernet 0/0, last update 1 min, CONFLICT
  Hops 64, Lifetime 1800 sec, AddrFlag=0, OtherFlag=0, MTU=1500
  Preference=Medium
  Reachable time 0 (unspecified) ms, Retransmit time 0 (unspecified) ms
  Prefix 2002::/64 on-link autoconfig
  Valid lifetime 8002, preferred lifetime 2008
```

- Use the **debug ipv6 nd [ar | dad | neighbor-state | packet [neighbor | router]]** command to enable debug messages for ND functions on this router. This command details the processing of ND messages and all resulting state changes and errors. To use this command, specify whether you will debug address resolution changes using the **ar** keyword, DAD events using the **dad** keyword, neighbor cache states using the **neighbor-state** keyword, ND packets using the **packet neighbor** parameter, or RA messages using the **packet router** parameter. To enable debug messaging for all ND DAD events, enter the command as follows:

#debug ipv6 nd dad



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

IPv6-Aware HTTP Server in AOS

In AOS firmware release 18.3.01, support for IPv6-aware Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS) servers was implemented in all AOS products that support the use of the Web-based graphical user interface (GUI). In previous versions of AOS firmware, the HTTP servers only recognized IPv4 packets, but with the new implementation of the IPv6-aware HTTP server, HTTP and HTTPS servers will respond to IPv6 HTTP requests. The following commands are available for use with the IPv6-aware HTTP(S) server.



*These commands are also used when configuring the IPv4 HTTP servers, except where specified using the **ipv6** keyword.*

1. Use the **http authentication** *<list name>* command to assign a specified authentication, authorization, and accounting (AAA) list to use in authentication to the AOS device's IPv6-aware HTTP server. AAA must be enabled globally (using the **aaa** command) before you can apply an AAA list to HTTP authentication. By default, no HTTP authentication is configured. Using the **no** form of this command removes the AAA list from HTTP authentication. To assign an AAA list, enter the command from the Global Configuration mode as follows:

```
(config)#http authentication Mylist1
```

2. Use the **http language** *<language>* command to specify the language used in the GUI. The *<language>* parameter can be **english**, **frenchcanadian**, **italian**, **latinamspanish**, or **simplifiedchinese**. Using the **no** form of this command returns the GUI language to the default value. To change the GUI language from the default of English, enter the command from the Global Configuration mode prompt as follows:

```
(config)#http language frenchcanadian
```

3. Use the **http secure-ciphersuite** [**aes128-sha** | **aes256-sha** | **des-cbc-md5** | **des-cbc-sha** | **des-cbc3-md5** | **des-cbc3-sha** | **dhe-rsa-aes128-sha** | **dhe-rsa-aes256-sha** | **edh-rsa-des-cbc-sha** | **edh-rsa-des-cbc3-sha** | **rc4-md5** | **rc4-sha**] command to enable a secure sockets layer (SSL) cipher suite. [Table 13](#) outlines the properties of the SSL parameters. By default, no cipher suites are enabled. Using the **no** form of this command removes the SSL cipher suite configuration.

Table 13. HTTP Secure Cipher Suite Parameters

Command Keyword	Description
aes128-sha	Enables an SSL version 3 (SSLv3) cipher suite with the following properties: Key exchange algorithm (Kx)=Rivest, Sharmir, and Dleman (RSA); Authentication (Auth)=RSA; Bulk encryption algorithm (E)=128-bit Advanced Encryption Standard (AES); Hash function (Hash)=secure hash algorithm 1 (SHA-1).
aes256-sha	Enables an SSLv3 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=256-bit AES; Hash=SHA-1.

Table 13. HTTP Secure Cipher Suite Parameters (Continued)

Command Keyword	Description
des-cbc-md5	Enables an SSL version 2 (SSLv2) cipher suite with the following properties: Kx=RSA; Auth=RSA; E=56-bit Data Encryption Standard (DES); Hash=message-digest algorithm (MD5).
des-cbc-sha	Enables an SSLv3 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=56-bit DES; Hash=SHA-1.
des-cbc3-md5	Enables an SSLv2 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=168-bit 3DES; Hash=MD5.
des-cbc3-sha	Enables an SSLv3 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=168-bit 3DES; Hash=SHA-1.
dhe-rsa-aes128-sha	Enables an SSLv3 cipher suite with the following properties: Kx=DH; Auth=RSA; E=128-bit AES; Hash=SHA-1.
dhe-rsa-aes256-sha	Enables an SSLv3 cipher suite with the following properties: Kx=DH; Auth=RSA; E=256-bit AES; Hash=SHA-1.
edh-rsa-des-cbc-sha	Enables an SSLv3 cipher suite with the following properties: Kx=DH; Auth=RSA; E=56-bit DES; Hash=SHA-1.
edh-rsa-des-cbc3-sha	Enables an SSLv3 cipher suite with the following properties: Kx=DH; Auth=RSA; E=168-bit 3DES; Hash=SHA-1.
rc4-md5	Enables an SSLv2 or SSLv3 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=128-bit Rivest Cipher 4 (RC4); Hash=MD5.
rc4-sha	Enables an SSLv3 cipher suite with the following properties: Kx=RSA; Auth=RSA; E=128-bit RC4; Hash=SHA-1.

To enable an SSL cipher suite, enter the command from the Global Configuration mode prompt as follows:

```
(config)#http secure-ciphersuite rc4-sha
```

4. Use the **http secure-server [allow-ssl2] [<TCP port>]** command to enable the HTTP secure server and default the server to use SSLv3. The optional **allow-ssl2** parameter specifies that the server will fall back to SSLv2 if it is unable to connect using SSLv3. The optional **<TCP port>** parameter specifies an alternate port for HTTPS connections. By default, the HTTP secure server is disabled. Using the **no** form of this command disables the HTTP secure server. To enable the HTTP secure server, enter the command from the Global Configuration mode prompt as follow:

```
(config)#http secure-server
```

5. Use the **http server [<TCP port>]** command to enable the HTTP server. The optional **<TCP port>** parameter specifies an alternate port for the HTTP server. By default, the HTTP server is disabled. Using the **no** form of this command disables the HTTP server. To enable the server, enter the command from the Global Configuration mode prompt as follows:

```
(config)#http server
```

6. Use the **http session-limit <number>** command to set the maximum number of HTTP sessions allowed. By default, the maximum number of allowed HTTP sessions is **100**. Valid range is **0** to **100** sessions. Using the **no** form of this commands returns the maximum allowed sessions to the default value. To change this value, enter the command as follows:

```
(config)#http session-limit 75
```

7. Use the **http session-timeout <value>** command to set the HTTP session timeout value. By default, the session times out after **600** seconds. Valid range is **10** to **86400** seconds. Using the **no** form of this command returns the session timeout to the default value. To change the session timeout, enter the command from the Global Configuration mode prompt as follows:

```
(config)#http session-timeout 1500
```

8. Use the **http source-interface <interface>** command to specify a source interface for HTTP traffic originated by the AOS unit. The IPv6 address of the specified interface is used to source all HTTP traffic. This command affects the HTTP client, and not the HTTP server. The **<interface>** parameter specifies the source interface for HTTP traffic. Interfaces are specified in the format **<interface type [slot/port | slot/port.subinterface id | interface id | interface id.subinterface id]**, for example, for an Ethernet subinterface, use **eth 0/1.1**; for a PPP interface, use **ppp 1**, etc. Type **http source-interface ?** for a complete list of available interfaces. By default, no HTTP source interface is defined. Using the **no** form of this command to remove the source interface definition. To specify a source interface for HTTP traffic, enter the command as follows from the Global Configuration mode:

```
(config)#http source-interface eth 1/1
```

9. Use the **http ipv6 [access-class <ipv6 acl name> in | secure-access-class <ipv6 acl name> in]** command to restrict access to the HTTP server using the specified IPv6 ACL. The **access-class** parameter specifies that the restriction applies to all self-bound HTTP connections, and the **secure-access-class** parameter specifies that the restriction applies to all self-bound HTTPS connections. The **<ipv6 acl name>** parameter specifies the IPv6 ACL to use (refer to [IPv6 ACLs on page 57](#) for more information about creating IPv6 ACLs). The **in** parameter specifies that the ACL is applied to incoming connections. By default, no ACLs are applied to HTTP connections. Using the **no** form of this command removes the IPv6 ACL from the HTTP connection. To specify HTTP access is restricted by a previously configured IPv6 ACL, enter the command from the Global Configuration mode as follows:

```
(config)#http ipv6 access-class ipv6acllist1
```


<vrf name> parameter specifies a nondefault VRF instance on which to define the IPv6 name server address. The *<ipv6 address>* parameter specifies the IPv6 address associated with the name server. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. By default, there are no name servers configured. Using the **no** form of this command removes the name server address. To add the name server's IPv6 address, enter the command as follows:

```
(config)#name-server 2001:DB8:1::1
```

3. Use the **domain-lookup** command to enable or disable IPv6 DNS client domain lookup. By default, domain lookup is enabled. To disable domain lookup, enter the **no** version of this command from the Global Configuration mode prompt as follows:

```
(config)#no domain-lookup
```

4. Use the **domain-name [vrf <vrf name>] <domain name>** command to define the default IPv6 domain name to be used by AOS to resolve host names. The optional **vrf <vrf name>** parameter specifies a nondefault VRF instance on which to define the IPv6 domain name. Each VRF instance has its own domain name. Specifying a VRF name in the command applies the domain name to the named VRF. Issuing the command without specifying a VRF applies the command the default unnamed VRF. The *<domain name>* parameter is a string of characters that is the domain name used to resolve unqualified host names. It is not necessary to include the initial period that separates the unresolved name from the default domain name when entering the domain name. By default, no domain name is defined. Using the **no** form of this command removes the domain name definition. To define the domain name for a device, enter the command from the Global Configuration mode as follows:

```
(config)#domain-name company1
```

5. Use the **domain-proxy [vrf <vrf name>] [failover]** command to enable or disable the IPv6 DNS proxy. Enabling the proxy allows the AOS device to act as a proxy for other units in the network. The optional **vrf <vrf name>** parameter specifies a nondefault VRF instance on which to enable the DNS proxy. The optional **failover** parameter enables IPv6 DNS failover mode on the domain proxy. By default, the DNS proxy is disabled. Using the **no** form of this command disables the proxy. To enable the DNS proxy, enter the command from the Global Configuration mode as follows:

```
(config)#domain-proxy
```

Troubleshooting IPv6 DNS

The following **show** and **debug** commands can be used to view IPv6 DNS statistics and to verify IPv6 DNS configuration on the AOS unit. These commands are executed from the Enable mode.

1. Use the **clear host [vrf <name>] [* | <hostname>]** command to clear dynamic entries from the IPv6 DNS host table. The optional **vrf <name>** parameter specifies a nondefault VRF on which to clear all the dynamic entries. If the VRF is not specified, entries are cleared on the default VRF. The ***** parameter clears all entries (both IPv4 and IPv6) from the DNS host table. The *<hostname>* parameter clears the specified entry from the DNS host table. Enter the command as follows to clear all DNS host table entries:

```
#clear host *
```

2. Use the **show hosts [vrf <name> | verbose | realtime]** command to display the contents of the DNS host table. Output from this command displays both IPv4 and IPv6 entries, as well as separate server addresses for the DNS client and proxy. The optional **vrf <name>** parameter specifies a nondefault VRF for which to display host table information. If a VRF is not specified, host table information is

displayed for the default VRF. The optional **verbose** parameter specifies that details of the IP domain name, style, name servers, and host table entries are displayed without truncation of long addresses and host names. The optional **realtime** parameter specifies that information is displayed in real time. To display DNS host table information, enter the command as follows:

#show hosts

Name/address lookup uses domain name service

DNS Proxy is enabled

Name servers are 10.23.115.254

Current proxy server is 10.23.115.254

Current client server is 10.23.115.254

Host	Flags	Age	Type	Priority	Address/Alias
abc.com	temp	193	A		-2000:ef0a::1500:37af:362:ed
Archive.msstate.edu	temp	16907	A		-130.18.80.18
dns11.11nwd.net	temp	673	A		-200:a50:1a0e::1500:eddf

- Use the **show name-server [vrf <name> | realtime]** command to display the current name server's IPv4 and IPv6 address and the source of its addresses. Address sources include DHCPv4, DHCPv6, PPP-IPCP, and user configured addresses. Also included in the command output are the current proxy and client server addresses. The optional **vrf <name>** parameter specifies a nondefault VRF for which to display name server address information. If a VRF is not specified, host table information is displayed for the default VRF. The optional **realtime** parameter specifies that information is displayed in real time. To display DNS name server address information, enter the command as follows:

#show name-server

Current	Name server address	Source
	2000:ef0a::1500:37ag:362:ed	DHCPv6
proxy -->	2000:a50:1a0e::1500:eddf	DHCPv6
	10.23.115.254	DHCPv4
client -->	192.168.101.1	PPP
	8.8.8.8	User
	8.8.4.4	User

- Use the **debug dns client** command to enable debug messages for the DNS client. The **no** form of this command disables debug messages for the DNS client. Output from this command includes IPv6 AAAA records. Enter the command as follows:

#debug dns client

- Use the **debug dns table** command to enable debug messages for DNS host table events. The **no** form of this command disables debug messages for host table events. Output from this command includes IPv6 AAAA records. Enter the command as follows:

#debug dns table

- Use the **debug dns proxy** command to enable debug messages for domain name proxy events. The **no** form of this command disables debug messages for the domain name proxy. Output from this command includes IPv6 AAAA and PTR records. Enter the command as follows:

#debug dns proxy



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

ICMPv6 in AOS

IPv6 supports an updated version of Internet Control Message Protocol (ICMP) called ICMPv6. ICMPv6 functions similarly to ICMP in IPv4, providing error messages for errors encountered in forwarding or delivery by the destination node or intermediate router, and informational messages that provide diagnostic functions and additional host functionality. For error messages in ICMPv6, the high order bit of the Type field is set to **0** (with a range of **0** to **127**) and for informational messages the high order bit of the Type field is set to **1** (with a range of **128** to **255**). Although ICMPv6 is similar to ICMPv4, the message types are similar but use different types and codes. [Table 14](#) outlines the messages types in both ICMPv4 and ICMPv6.

Table 14. ICMPv4 and ICMPv6 Message Types

ICMPv4 Message	ICMPv6 Equivalent
Destination Unreachable-Network Unreachable (Type 3, Code 0)	Destination Unreachable-No route to Destination (Type 1, Code 0)
Destination Unreachable-Protocol Unreachable (Type 3, Code 2)	Parameter Problem-Unrecognized Next Header Field (Type 4, Code 1)
Destination Unreachable-Port Unreachable (Type 3, Code 3)	Destination Unreachable-Port Unreachable (Type 1, Code 4)
Destination Unreachable-Fragmentation Needed and DF Set (Type 3, Code 4)	Packet Too Big (Type 2, Code 0)
Time Exceeded-TTL Expired (Type 11, Code 0)	Time Exceeded-Hop Limit Exceeded (Type 3, Code 0)
Redirect (Type 5, Code 0)	ND Redirect Message (Type 137, Code 0)

The new addition to traditional ICMP is that in ICMPv6 a framework is provided for maintenance functions such as MLD, ND, and Mobile IPv6. ICMPv6 packets are identified by a 58 in the Next Header field of the IPv6 packet.

Troubleshooting ICMPv6

The following **debug** commands can be used to enable ICMPv6 debug messages in the AOS unit. These commands are executed from the Enable mode.

1. Use the **debug ipv6 icmp [send | rcv]** command to enable debug messages that show ICMPv6 packets being sent or received by the local IPv6 stack. These messages do not include ICMP packets being exchanged between other devices because those packets appear as IPv6 packets to the local router. Using the **send** or **rcv** keywords limits the debug messages to ICMP messages sent or received by the IPv6 stack. The **no** form of this command disables debug messages for ICMPv6. To enable ICMPv6 debug messaging, enter the command as follows:

#debug ipv6 icmp

```
ICMPv6 SEND: To [2001:DB8:8967::10] Type=128 Code=0 Length=108 Details:echo request
id=0036 seq=0001
ICMPv6 SEND: Source changed to [2001:DB8:8967:1::100] before transmit
ICMPv6 RECV: From [1001:DB8:8967::10] to [2001:DB8:8967:1::100] [eth 0/1]
Type=129 Code=0 Length=108 Details: echo reply
id=0036 seq=0001
```



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

NTP over IPv6

NTP over IPv6 functions similarly to NTP over IPv4, and can be used for Network Quality Monitoring (NQM) or instead of Simple Network Time Protocol (SNTP) to synchronize the AOS product clock. Configuration of NTP over IPv6 in AOS products consists of specifying an IPv6 address for the NTP peer or an NTP server of the AOS product. The commands used for NTP over IPv6 are discussed in the following sections.

Specifying an NTP Peer

NTP will be enabled when an IPv6 NTP peer is configured. Use the **ntp peer** *<hostname | ipv6 address>* [*<interface>*] [**maxpoll** *<value>*] [**minpoll** *<value>*] [**normal-sync** | **prefer** | **source** *<interface>*] [**version**] command to specify a peer association with another NTP system and configure its parameters. Any combination of peer associations can be configured simultaneously. The *<hostname | ipv6 address>* parameter specifies the host name or IPv6 address associated with the NTP peer. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The *<interface>* parameter, which is available when an IPv6 link-local address is used, specifies which interface NTP should use to send NTP requests. Interfaces must be specified when using an IPv6 link-local address for the NTP peer. Specify interfaces in the *<interface>* *<slot/port | interface id>* format. The optional **maxpoll** *<value>* parameter specifies the maximum polling interval for NTP packets, in seconds as a power of two. The allowable range is **4** to **17**. For example, setting the **maxpoll** to **10** indicates a maximum polling interval of **1024** seconds. The optional **minpoll** *<value>* parameter specifies the minimum polling interval for NTP packets, in seconds as a power of two. The allowable range is **4** to **17**. For example, setting the **minpoll** value to **6** indicates a minimum polling interval of **64** seconds. The optional **normal-sync** parameter specifies that rapid synchronization is disabled. The optional **prefer** parameter specifies the preference of using the specified peer above all other configured NTP peers. The optional **source** *<interface>* parameter specifies the physical or virtual source interface to use for the peer. The optional **version** parameter specifies the version number for outgoing NTP packets. Valid version range is **2** to **4** for NTP; however, only version 4 is allowed when specifying an IPv6 peer address. These optional parameters can be executed in any combination to obtain the desired configuration. By default, IPv6 NTP does not have a peer set. Once it is enabled, the default version is **4**, the default **minpoll** interval is **6** (64 seconds), and the default **maxpoll** interval is **10** (1024 seconds). Using the **no** form of this command returns to the default setting.

To enable IPv6 NTP and specify an NTP peer, enter the command from the Global Configuration mode as follows:

```
(config)#ntp peer fe80::2 vlan 1
```

Specifying an NTP Server

You can also enable NTP by specifying the NTP server. Use the **ntp server** *<hostname | ipv6 address>* [*<interface>*] [**maxpoll** *<value>* | **minpoll** *<value>* | **normal-sync** | **prefer** | **source** *<interface>* **version**] command to specify a server association with another NTP system and configure its parameters. Any combination of server associations can be configured simultaneously. The *<hostname | ipv6 address>* parameter specifies the host name or IPv6 address associated with the NTP server. IPv6 addresses should be expressed in colon hexadecimal format (**X:X:X:X::X**), for example, **2001:DB8:1::1**. The *<interface>* parameter, which is available when an IPv6 link-local address is used, specifies which interface NTP should use to send NTP requests. Interfaces must be specified when using an IPv6 link-local address for the NTP server. Specify interfaces in the *<interface>* *<slot/port | interface id>* format. The optional **maxpoll** *<value>* parameter specifies the maximum polling interval for NTP packets, in seconds as a power of two. The allowable range is **4** to **17**. For example, setting the **maxpoll** to 10 indicates a maximum polling interval of **1024** seconds. The optional **minpoll** *<value>* parameter specifies the minimum polling interval for NTP packets, in seconds as a power of two. The allowable range is **4** to **17**. For example, setting the **minpoll** value to 6 indicates a minimum polling interval of **64** seconds. The optional **normal-sync** parameter specifies that rapid synchronization is disabled. The optional **prefer** parameter specifies the preference of using the specified server above all other configured NTP servers. The optional **source** *<interface>* parameter specifies the physical or virtual source interface to use for the server. The optional **version** parameter specifies the version number for outgoing NTP packets. Valid version range is **2** to **4** for NTP; however, only version 4 is allowed when specifying an IPv6 server address. These optional parameters can be executed in any combination to obtain the desired configuration. By default, IPv6 NTP does not have a server set. Once it is enabled, the default version is **4**, the default **minpoll** interval is **6** (64 seconds), and the default **maxpoll** interval is **10** (1024 seconds). Using the **no** form of this command returns to the default setting.

To enable IPv6 NTP and specify an NTP server, enter the command from the Global Configuration mode as follows:

```
(config)#ntp server fe80::2 vlan 1
```

Viewing IPv6 NTP Associations

You can view active NTP associations by entering the **show ntp associations** command from the Enable mode. This command displays the IPv6 (or IPv4) address, reference ID, and other information about NTP associations. Associations with IPv6 addresses are displayed on two lines due to their length. To display NTP associations, enter the command as follows:

```
>enable
```

```
#show ntp associations
```

Address	ref ID	st	when	poll	reach	delay	offset	disp
~172.22.47.47	209.81.9.7	2	226	64	01	1.79	462767.44	7937.74
*~fe80::20a:1ff:fe02:6936	2139029760	4	15	64	017	1.34	-46.82	064

*master (syncd), # master (unsyncd), + selected, - candidate, ~configured, x falseticker, . excess

IPv6 Traffic Control in AOS

IPv6 traffic control can be achieved in AOS by using access control lists (ACLs), access control policies (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 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)#
```



If you are using both IPv4 and IPv6 ACLs, they must have different names. You cannot assign an IPv4 ACL and also 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 are 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.



*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 three following 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::/464**. 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)#
```



If you are using both IPv4 and IPv6 ACLs, they must have different names. You cannot assign an IPv4 ACL and also 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 are 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.
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 flag 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 additionally need to enter a message code. Message code range is **0** to **255**.
Named ICMPv6 messages include the following:

Table 15. 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.
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 multicast 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 NA message.
nd-ns	Indicates an ND 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.

Table 15. ICMPv6 Message Names (Continued)

Message Name Entry	Message Meaning
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 an RS message.
time-exceeded	Indicates the time limit has been exceeded.
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 you to use 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 Trivial File Transfer Protocol (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 InboundIPv4 in
(config)#
```

For more information about specific commands used in applying either IPv4 or IPv6 ACLs to specific features, refer to the *AOS Command Reference Guide*, available online at <https://supportforums.adtran.com>.

IPv6 Access Control Policies

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 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. The ACP named must be an IPv6 ACP.
 - 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 up remote administrative access to the unit (Telnet, SSH, ICMP, HTTP, 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 easily recreated.
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 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. The ACP named must be an IPv6 ACP.
- 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 the maximum number of allowed policy sessions in the AOS product for both IPv4 and IPv6 combined, 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
```


- 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
```

- 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
```

Configuring IPv6 Traffic Control on 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.

- 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
```

- 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.*

Troubleshooting IPv6 ACLs and ACPs

The following **clear** and **show** commands can be used to clear IPv6 ACL or ACP entries and statistics and to view ACL and ACP configuration information in the AOS unit. These commands are executed from the Enable mode.

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

#clear ipv6 access-list

2. 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 acl name>* [**ahp** | **esp** | **gre** | *<protocol>*] *<source ipv6 address>* *<destination ipv6 address>*
- **clear ipv6 policy-session** [**vrf** *<name>*] *<ipv6 acl name>* [**tcp** | **udp**] *<source ipv6 address>* *<source port>* *<destination ipv6 address>* *<destination port>*
- **clear ipv6 policy-sessions** [**vrf** *<name>*] *<ipv6 acl 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

- 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 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. 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
```

7. 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
```

Configuring IPv6 Traffic to Traverse an IPv4 GRE Tunnel

In AOS firmware release R10.1.0, the capability to tunnel IPv6 packets over an IPv4 Generic Route Encapsulation (GRE) tunnel was incorporated into AOS. This feature can be used as a temporary or cost-effective method of allowing IPv6 hosts on two different networks to communicate with each other across the IPv4 wide area network (WAN) or Internet. Although communication would be best served by an end-to-end IPv6 connection, encapsulating IPv6 packets within an IPv4 GRE tunnel does provide several benefits, such as the ability to secure IPv6 site-to-site traffic with IPsec and providing a method for IPv6 traffic to traverse VRF instances. The feature functions by encapsulating IPv6 payloads into IPv4 GRE tunnel packets at the egress point and decapsulating the tunnel packets at the ingress point to then pass on the original IPv6 payload to the appropriate destination.

In order to accommodate this feature, the tunnel interface now supports IPv6 addressing, IPv6 egress routes, ND (including DAD and SLAAC), ICMPv6 error generation and error handling, and DHCPv6 relay and server configurations. In addition, the tunnel interface supports the dual IPv4/IPv6 stack, which allows the interface to have both IPv4 and IPv6 addresses. Because of this, both types of traffic can be encapsulated into an IPv4 GRE tunnel, and sent between the same tunnel endpoints. You can configure multiple tunnels if you want to enable different GRE features (such as checksum) on different tunnels for different types of tunnel traffic. The tunnel key is used to distinguish various tunnels on the interface, as well as to configure routes properly. Each tunnel is assigned only the IP address type that is applicable.

IPv6 over IPv4 GRE Tunnel Packet Headers

The following figures describe the IPv4 GRE packet header when transporting an IPv6 payload, and also the GRE packet header when used with an IPv6 payload.

The IPv6 over IPv4 GRE packet has the following format:

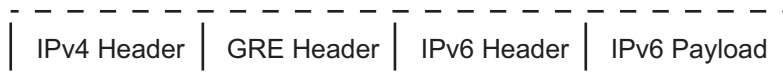


Figure 5. IPv6 over IPv4 GRE Packet Headers

The GRE packet itself appears in this format:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
C	K	S	Reserved0						Ver	Protocol Type																					
Checksum (Optional)										Reserved1 (Optional)																					
Key (Optional)																															
Sequence Number (Optional)																															

Figure 6. GRE Packet Header

The only difference in the GRE packet when sending an IPv4 payload versus an IPv6 payload is in the Protocol Type header field. When sending an IPv4 payload, the Protocol Type header value is 0x800, but when sending an IPv6 payload the value is 0x86DD.

Considerations for Transporting IPv6 Packets in an IPv4 GRE Tunnel

The following sections describe some considerations for transporting IPv6 packets in an IPv4 GRE tunnel. These considerations include feature functions like maximum transmission unit (MTU) settings, packet hop limits and time-to-live (TTL) settings, traffic class and type of service (ToS) settings, ND behavior, EUI-64 IPv6 addressing behavior, and traffic source validation.

Tunnel MTU and Fragmentation

The MTU settings will play a large part in the effectiveness of transporting IPv6 packets over the IPv4 GRE tunnel. Ideally, the MTU value should be set low enough that once encapsulation overhead is added, the new packet does not need to be fragmented. MTU calculations depend on the MTU of the egress interface (the physical interface that the tunnel packets exit), the tunnel mode used, and the GRE features enabled.

IPv4 path MTU (PMTU) discovery is not currently supported. Instead, the default IPv4 tunnel MTU of GRE tunnel interfaces is calculated by subtracting **20** bytes (for the IPv4 header overhead) from the IPv4 MTU of the egress interface, and then subtracting the GRE header overhead. GRE overhead varies, based on the configured features of the tunnel key, checksum, and sequence numbers, and will be a minimum of **4** bytes (default) up to a maximum of **16** bytes (4 bytes for each configured feature). For example, a 1500 byte MTU on the physical interface with no added GRE features yields a default tunnel MTU of 1476 bytes.

The tunnel interface has an IPv6 address for this feature and possibly an IPv4 address. The egress interface will have an IPv4 address. The default IPv6 tunnel MTU for the GRE tunnel interface is calculated as described previously (using the IPv4 MTU of the egress interface as the basis), except that the MTU will never be set below **1280** bytes. If the MTU is set to below 1280 bytes, a warning is generated. If the tunnel interface has both an IPv4 and IPv6 address, the default calculated values for both the IPv4 MTU and the IPv6 MTU must be equal. You can set either MTU value independently using the **ip mtu** or **ipv6 mtu** commands for IPv4 or IPv6 MTU values respectively.

If an IPv6 packet is routed out of the tunnel, but is greater than the tunnel MTU, an ICMPv6 *Packet Too Big* message is generated and sent to the initiating host. The host can then lower the packet size so that no IPv4 fragmentation of the encapsulated packets occurs on the AOS product. It is still possible that some routers in the tunnel's path may have to fragment the IPv4 packets if a lower IPv4 MTU is encountered, so the don't fragment (DF) bit in the IPv4 header is cleared when carrying IPv6 payloads.

In addition, the IPv4 GRE tunnel packet (carrying the IPv6 payload) can be sent over an IPv4 IPsec tunnel. When using an IPv4 IPsec tunnel, more overhead is added to the packets. This additional overhead causes IPv4 fragmentation on the AOS unit before encryption and authentication is applied, and before the packet is encapsulated in another IPv4 header and ESP/AH headers. In this case, the GRE tunnel interface MTU should be manually overwritten (using the **ipv6 mtu** command) to take into account the additional IPsec overhead, which varies based on packet size and encryption or authentication algorithms. A recommended value of **1400** bytes accommodates most additional IPsec overhead.

Because the IPv4 MTU on a tunnel interface is **1500** bytes, the IPv6 MTU maximum is set to **1592** bytes. This setting allows the entire path MTU between two IPv6 hosts to be 1592 bytes (except that the physical interface MTU between the two tunnel endpoints is smaller) but prevents the hosts from having to fragment since PMTU discovery sees no MTU change. You should typically only override the IPv6 MTU with a value lower than the calculated default. If a larger value is entered, a warning is generated.

Hop Limit and TTL

The GRE tunnel is a single hop from the IPv6 perspective, and thus it functions much like a point-to-point link for the original IPv6 traffic inside the tunnel. The encapsulator and decapsulator each decrement the IPv6 hop limit by **1** before forwarding the original IPv6 packets. The encapsulator sets the IPv4 TTL field to **255**, just as is done for IPv4 packets over an IPv4 GRE tunnel.

Traffic Class and ToS

In traditional GRE tunneling, the GRE tunnel copies the ToS byte from the inner IPv4 header to the outer IPv4 header when encapsulating packets, and it ignores the IPv4 header ToS value when decapsulating the packet. This method leaves the IPv4 header with the original ToS value. When carrying an IPv6 payload, the GRE tunnel copies the traffic class value in the original IPv6 packet into the IPv4 ToS field to preserve any DSCP values in the traffic class field. When the packet is decapsulated, the IPv4 header ToS value is ignored, which leaves the IPv6 header with the original traffic class value.

When sequence numbers are used in the GRE tunnel, there can be a negative side effect. If two different kinds of traffic, with different ToS values, use the same tunnel interface and egress the same physical interface where a QoS map separates the traffic into different queues, the tunnel packets can be reordered by QoS before egressing the interface. The tunnel endpoint that receives the out-of-order packets will drop the packets that do not have the next expected GRE sequence number, if sequence numbers are being used. Therefore, sequence numbers should not be used when QoS is expected to reorder packets on the egress interface.

ND in GRE Tunnels

ND is supported over the tunnel. DAD and neighbor unreachability detection (NUD) are both used and supported over the tunnel, and the tunnel endpoints both accept and respond to NUD probe packets. However, AOS units do not send a NUD probe packet over the tunnel since there is no destination cache. Tunnel endpoint bidirectional reachability is achieved by using GRE keepalives.

EUI-64 Formatted Addresses

IPv6 link-local address interface IDs are based on the system MAC address in EUI-64 format. This applies to all EUI-64 formatted addresses, including the default link-local address, any configured link-local and EUI-64 addresses, and any autoconfigured addresses based on prefixes received in router advertisements. If a duplicate hardware address is found, the tunnel interface remains inoperable until a change is made to correct the address.

IPv6 Source Address Validation

Similarly to the validation of IPv4 packets, no source address validation of IPv6 original packets is performed when the packets exit the tunnel. However, the IPv6 firewall can be enabled to perform IPv6 source address validation. The firewall can ensure that packets with multicast addresses (FF00::/8), loopback addresses (::1), or IPv4-mapped IPv6 addresses (::FFFF:0:0/96) are silently discarded.

Configuring the IPv4 GRE Tunnel for an IPv6 Payload

The following sections describe the tunnel configuration for IPv6 payloads. All configuration is completed using the CLI. The following steps are necessary to configure this feature:

1. Create the tunnel, specify the tunnel mode, and enter the tunnel's configuration mode.
2. Enable IPv6 on the tunnel interface and create an IPv6 address for the interface.
3. Specify the MTU for the IPv6 packets on the tunnel interface.
4. Configure the tunnel source and destination IPv4 addresses.
5. Optionally, configure additional GRE parameters such as keepalives, tunnel keys, and checksum verification.
6. Configure the tunnel routing information.

Step 1: Creating the Tunnel

GRE tunnel interfaces are virtual interfaces created within the AOS software. Each tunnel interface must have a numerical identifier label. To create a GRE tunnel, specify the tunnel mode, and enter the tunnel's configuration mode, enter the **interface tunnel <number> [gre ip]** command from the Global Configuration mode. The *<number>* parameter of this command specifies the tunnel's numerical label identifier. Valid range is **1** to **1024**. The optional **gre ip** parameter of this command specifies that the tunnel is in GRE mode (default), and that is an IPv4 tunnel. When this parameter is used and a new tunnel interface is being created, the parameter creates the tunnel interface, specifies that all traffic (both IPv4 and IPv6) is encapsulated in an IPv4/GRE delivery header, and enters the tunnel's configuration mode. If the **gre ip** parameter is NOT used and the tunnel interface has not been previously created, an error is generated because the tunnel mode must be specified when creating a new tunnel interface. If the **gre ip** parameter is NOT used and the tunnel interface has been previously created, the command enters the tunnel's configuration mode. Using the **no** form of this command removes the tunnel interface.

For example, to create a new tunnel interface, specify the tunnel mode as GRE, specify that all traffic is encapsulated in the IPv4 GRE tunnel, and enter the tunnel's configuration mode, enter the command as follows:

```
(config)#interface tunnel 1 gre ip  
(config-tunnel 1)#
```

Step 2: Enabling IPv6 and Assigning IPv6 Addresses on the Tunnel Interface

After the tunnel has been created, you must enable IPv6 on the tunnel interface and assign an IPv6 address to the tunnel. The tunnel has to be configured like an IPv6 interface for this feature to function, and can be achieved by enabling IPv6 and receiving an automatically generated link-local address (using the **ipv6** command), or by applying an IPv6 address to the interface (using a variation of the **ipv6 address** command).

For example, to enable IPv6 on the tunnel interface, and assign an automatically generated link-local address, enter the **ipv6** command on the tunnel interface as follows:

```
(config-tunnel 1)#ipv6  
(config-tunnel1 )#
```


Step 3: Specifying the MTU Settings

Once the tunnel interface has been created, and configured for IPv6 capability, you must specify the MTU settings for IPv6 packets on the interface using the **ipv6 mtu** *<bytes>* command from the Tunnel Interface Configuration mode. This command sets the MTU and overrides the default calculated MTU. The *<bytes>* parameter specifies the largest number of bytes that can be transmitted through the interface. The minimum MTU size for IPv6 packets is **1280** bytes. The maximum size for a tunnel interface is **1592** bytes. By default, the MTU size for IPv6 packets is calculated as described in [Tunnel MTU and Fragmentation on page 70](#).

To specify the MTU value for IPv6 packets on the tunnel interface, enter the command as follows:

```
(config-tunnel 1)#ipv6 mtu 1400
(config-tunnel 1)#
```



You should typically only override the IPv6 MTU with a value lower than what is calculated by default. If a larger value is entered, a warning is generated.

Step 4: Configuring the Tunnel Source and Destination IPv4 Address

After configuring the tunnel interface and specifying the MTU settings, you should configure the tunnel's source and destination IPv4 address. The GRE tunnel source address is the IPv4 address, or interface name, for the public connection on the local router. Tunnel source addresses are used for the source address of the IPv4 packet delivery header and should be addresses assigned to you by your service provider. If you choose to use an interface (such as **eth 0/1**), the IPv4 address assigned to that interface is placed in the source address of the delivery header.



*If you use an interface (such as **eth 0/1**) for the configured tunnel source address, the specified interface must be active for the tunnel to negotiate. If the specified interface is not active, no data will be passed over the GRE tunnel.*

To set the tunnel's source address, enter the **tunnel source** [*<ipv4 address>* | *<interface>*] from the Tunnel Interface Configuration mode. The *<ipv4 address>* parameter specifies the IPv4 address used as the tunnel's source. IPv4 addresses should be expressed in dotted decimal notation (for example, **X.X.X.X**). The *<interface>* parameter specifies an interface to use as the tunnel's source. Interfaces are specified in the format *<interface type [slot/port | slot/port.subinterface.id | interface id | interface.id.subinterface id]>*. For a list of appropriate interfaces, enter **tunnel source ?** at the prompt. Using the **no** form of this command removes the source from the tunnel's configuration. For example, to specify the tunnel's source is IPv4 address **65.196.82.14**, enter the command as follows:

```
(config-tunnel 1)#tunnel source 65.196.82.14
(config-tunnel 1)#
```

The GRE tunnel destination address is the IPv4 address for the public connection on the remote router. Tunnel destination addresses are used for the destination address of the IPv4 packet delivery header and should be addresses assigned for the remote router by the service provider. Enter the **tunnel destination** `<ipv4 address>` from the Tunnel Interface Configuration mode to specify the tunnel's destination. The `<ipv4 address>` parameter specifies the IPv4 address of the tunnel's destination. IPv4 addresses should be expressed in dotted decimal notation (for example, **X.X.X.X**). To specify the tunnel's destination, enter the command as follows:

```
(config-tunnel 1)#tunnel destination 209.68.45.10
(config-tunnel 1)#
```

Step 5: Configure Additional GRE Tunnel Parameters (Optional)

Once you have configured the tunnel, enabled IPv6 on the interface, specified the MTU settings, and set the tunnel's source and destination, you can optionally configure keepalive messages, tunnel keys, checksums, and sequence numbers for the GRE tunnel. These options are not mandatory for the tunnel to function, but they can be useful for various networks. Keep in mind, however, that each of these options increases the MTU value necessary for the IPv4 GRE tunnel traffic.

To enable periodic status messages (keepalive messages) for the GRE tunnel, enter the **keepalive** `[<value>] [<number>]` command from the Tunnel Interface Configuration mode. By default, keepalive messages are disabled. Use the **no** form of this command to disable GRE keepalives. The optional `<value>` parameter specifies the time interval (in seconds) between transmitted keepalive packets. Valid range is **1** to **32767** seconds. By default, keepalive messages are sent every **10** seconds. The optional `<number>` parameter specifies the number of times to retry after failed keepalives before determining that the tunnel endpoint is down. Valid range is **1** to **255** retries. By default, the retry count is **3** times. To enable keepalive messages on the tunnel interface, enter the command as follows:

```
(config-tunnel 1)#keepalive
(config-tunnel 1)#
```

To enable a tunnel key, a value shared by both endpoints of the tunnel that provides minimal security and delineates between tunnels with the same source and destination, enter the **tunnel key** `<value>` from the Tunnel Interface Configuration mode. The `<value>` parameter specifies the key value for this tunnel. Valid range is **1** to **4294967294**. By default, tunnel keys are disabled. Use the **no** form of this command to remove the tunnel key. When enabled, the tunnel key information is stored in the GRE header and the key present bit is set to indicate to the receiver that a tunnel key was used. A matching tunnel key value must be used on both ends of the tunnel, otherwise received tunnel packets are discarded.



Tunnel keys do not provide security for the tunnel. Key information is sent (in clear text) in each GRE packet. To secure the tunnel information, you must use IPsec.

To configure a tunnel key, enter the command as follows:

```
(config-tunnel 1)#tunnel key 3546
(config-tunnel 1)#
```

To enable verification of the checksum of incoming GRE packets, and to include a checksum on outgoing tunnel packets, enter the **tunnel checksum** command from the Tunnel Interface Configuration mode. Checksums help recognize unintentional errors created in data during transmission. By default, checksums

are disabled. When enabled, however, the tunnel checksum is calculated for each outgoing GRE packet. The calculated result is stored in the GRE header. The checksum present bit is set in the GRE header to indicate to the router at the other end of the tunnel that a checksum was used. Both ends of the tunnel must have checksum enabled for proper operation. When both endpoints have checksum enabled, a packet with an incorrect checksum is dropped. If the endpoints differ in their checksum configurations, all packets flow without checksum verification. Use the **no** form of this command to disable tunnel checksums. To enable checksums, enter the command as follows:

```
(config-tunnel 1)#tunnel checksum
(config-tunnel 1)#
```

To enable sequence number checking on incoming GRE packets, enter the **tunnel sequence-datagrams** command from the Tunnel Interface Configuration mode. This command allows the tunnel to drop packets that arrive out of order. Both ends of the tunnel must have sequence numbering enabled. If the endpoints differ in their sequence numbering configuration, all packets will flow without any sequence number verification. Use the **no** form of this command to disable sequence number checking. By default, sequence number checking is disabled. To enable the feature, enter the command as follows:

```
(config-tunnel 1)#tunnel sequence-datagrams
(config-tunnel 1)#
```



Use sequence number verification cautiously. Out-of-order sequencing can occur due to network conditions outside of the tunnel endpoints, or if QoS maps are applied, and it can be difficult to establish a successful traffic flow after an out-of-order condition occurs.

Step 6: Configuring the Tunnel Routing Information

AOS products support various routing protocols including static routes, Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Border Gateway Protocol (BGP). RIP, OSPF, and BGP are all routing protocols that allow routers to share the information contained in their route tables with other routers in the network. These routing protocols are generally used on networks that frequently change or contain a large number of nodes. For small applications, manually adding static routes to the router's IPv4 route table is the easiest method of configuration.

Manually adding static routes to the IPv4 route table requires two steps: determining the routes needed (destination address and subnet mask, next-hop address or forwarding interface, and default route) and adding the route to the IPv4 route table. To complete these steps, enter the **ip route** *<ipv4 address>* *<subnet mask>* [*<interface>* | *<ipv4 address>*] [*<administrative distance>*] command from the Global Configuration mode. The *<ipv4 address>* parameter specifies the IPv4 network address to add to the IPv4 route table. IPv4 addresses should be expressed in dotted decimal notation (for example, **X.X.X.X**). The *<subnet mask>* parameter specifies the subnet mask that corresponds to a range of IPv4 address or a specific host. Subnet masks can be expressed in dotted decimal notation (for example, **255.255.255.0**) or as a prefix length (for example, **/24**). The *<interface | ipv4 address>* parameter specifies the far-end IPv4 addresses or an egress interface in the AOS unit. Enter **ip route** *<ipv4 address>* *<subnet mask>* **?** command to display a complete list of available egress interfaces. The optional *<administrative distance>* parameter specifies an administrative distance associated with a particular router used to determine the best route when multiple routes to the same destination exist. The lower the administrative distance, the more preferable the route. Valid range is **1** to **255**. Use the **no** form of this command to remove the route from the route table.

To add a static route for the tunnel interface, enter the command as follows:

```
(config)#ip route 192.168.2.0 255.255.255.0 tunnel 1
(config)#ip route 0.0.0.0 0.0.0.0 64.128.32.16
```



An IPv6 route can also be manually entered into the IPv6 route table using the **ipv6 route** command. Refer to [Adding Static Routes to the IPv6 Route Table on page 16](#) for more information about how to use the **ipv6 route** command.

IPv6 over IPv4 GRE Tunnel Configuration Example

The following example illustrates how to configure the IPv4 GRE tunnel to carry IPv6 traffic over an IPv4 network. This example should be used for illustrative purposes only. You will need to make configuration changes to ensure the configuration will function in your network.

In this scenario, an IPv6 packet of 1500 bytes carrying an ICMPv6 echo-request payload is initiated from 2001:DB8:1::1 on Host Y, and is sent to 2001:DB8:2::2 on Host Z. Upon reaching Router A, the packet's route lookup deems that it should be sent out the interface tunnel 1. Since the default calculated IPv6 MTU on tunnel 1 is only 1476 bytes, a *Packet Too Big* ICMPv6 error is sent back to Host Y. Host Y fragments the original IPv6 packet into two fragments, the first of which is now 1476 bytes and is sent to Router A. The packet egresses the tunnel interface, and is encapsulated into a 4-byte GRE header with no optional features, and an outer IPv4 header from 209.68.45.10 to 65.196.82.14 with the original IPv6 traffic class byte copied into the IPv4 ToS byte, a TTL of 255, and the DF bit cleared. This new IPv4 packet from Router A egresses through interface ppp 1, due to the IPv4 default route of Router A. Once the packet reaches Router B, the local stack matches the reversed source and destination addresses from the IPv4 header to interface tunnel 1, and sends it to the tunnel interface. The packet is decapsulated, removing the IPv4 and GRE headers. The original IPv6 fragmented packet is sent out the LAN interface based on a connected route for the 2001:DB8:2::/64 network. Host Y transmits the second fragment similarly through the network. Host Z then reassembles the fragments and sends an ICMPv6 echo reply that returns through the network over the GRE tunnel in the same way the request was received. [Figure 7](#) displays the network topology in which an IPv4 GRE tunnel is used to carry IPv6 traffic from host Y over the IPv4 network to Host Z.

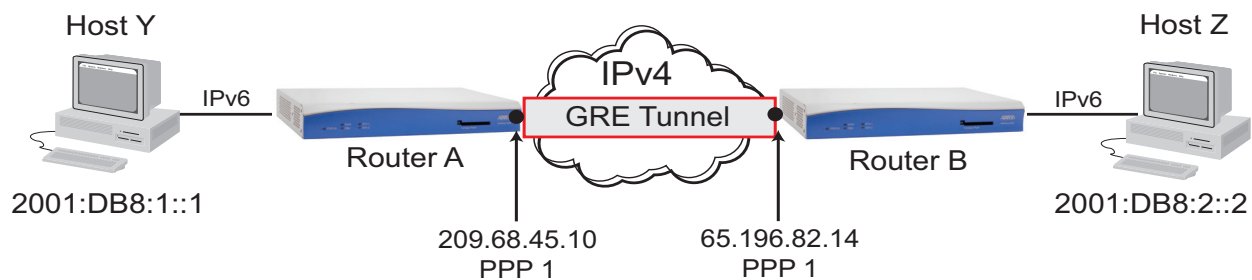


Figure 7. IPv6 over IPv4 GRE Tunnel Network Topology

The following is the sample configuration for both Router A and Router B:

Router A Configuration:

```
!  
ipv6 unicast-routing  
!  
interface ppp 1  
    ip address 209.68.45.10 255.255.255.0  
    no shutdown  
!  
interface tunnel 1 gre ip  
    no ip address  
    ipv6  
    ipv6 address 2001:DB8:1000::1/64  
    tunnel source 209.68.45.10  
    tunnel destination 65.196.82.14  
    no shutdown  
!  
ip route 0.0.0.0 0.0.0.0 ppp 1  
!  
ipv6 route 2001:DB8:2::/64 tunnel 1  
!
```

Router B Configuration:

```
!  
ipv6 unicast-routing  
!  
interface ppp 1  
    ip address 65.196.82.14 255.255.255.0  
    no shutdown  
!  
interface tunnel 1 gre ip  
    no ip address  
    ipv6  
    ipv6 address 2001:DB8:1000::2/64  
    tunnel source 65.196.82.14  
    tunnel destination 209.68.45.10  
    no shutdown  
!  
ip route 0.0.0.0 0.0.0.0 ppp 1  
!  
ipv6 route 2001:DB8:1::/64 tunnel 1  
!
```

Troubleshooting the Tunnel Interface

The following **show** and **debug** commands can be useful in troubleshooting your tunnel interface configuration. These commands are entered from the Enable mode.

1. Use the **show ipv6 interfaces tunnel** *<number>* command to display the configuration information and status of a specific tunnel interface. The *<number>* parameter specifies a tunnel interface. Valid range is **1** to **1024**. Enter the command as follows:

```
#show ipv6 interfaces tunnel 5
```

2. Use the **debug interface tunnel** *<number>* command to enable debug messaging for the tunnel interface. These messages provide status changes and details about packets processed over the tunnel, including the encapsulation, forwarding, and decapsulation of the IPv6 packet sent using the IPv4 GRE tunnel. The **no** form of this command disables debug messaging for the interface.



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

The following is sample output from the **debug interface tunnel** command:

```
#debug interface tunnel 1
```

```
2011.03.18 09:31:49 TUNNEL.1 Encapsulating original packet 2001:DB8:1::1-  
>2001:DB8:2::2 (len=100 hop limit=64)
```

```
2011.03.18 09:31:49 TUNNEL.1 Forwarding GRE packet 209.68.45.10-  
>65.196.852.14 (len=124 ttl=255)
```

```
2011.03.18 09:31:57 TUNNEL.1 Decapsulating GRE packet 65.196.82.14-  
>209.68.45.10 (len=124 ttl=253)
```

```
2011.03.18 09:31:57 TUNNEL.1 Forwarding original packet 2001:DB8:2::2-  
>2001:DB8:1::1 (len=100 hop limit=63)
```

Additional Resources

The following tables list additional resources for IPv6 information. The information in *Table 16* is provided by the Internet Engineering Task Force (IETF). The information in *Table 17* is additional IPv6 documentation provided by ADTRAN, available online at <https://supportforums.adtran.com>.

Table 16. IETF IPv6 Resources

Subject	Request for Comments (RFC) Article
IPv6 Addressing	RFC 4291
IPv6 Temporary Address/Interface IDs	RFC 4941
IPv6 Packet Headers	RFC 6397
IPv6 Global Unicast Addresses	RFC 3587
IPv6 Site-Local Address Replacement	RFC 1918
IPv6 Neighbor Discovery	RFC 4861
IPv6 Packets over Ethernet	RFC 2464
ICMPv6	RFC 4443
DHCPv6	RFC 3315, 4477
DHCPv6 Prefix Delegation	RFC 3633
DNS Enhancements for IPv6	RFC 3596
OSPFv3 for IPv6	RFC 1253, 5643, 3411, 5340, 5838, 4552
IPv6 NTP	RFC 5908

Table 17. 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 Quality of Service	Configuring QoS in AOS
IPv6 VRRPv3	IPv6 VRRPv3 for AOS
OSPFv3	OSPFv3 in AOS
IPv6 Firewall	IPv6 Firewall Protection in AOS
RapidRoute in AOS	Configuring RapidRoute in AOS