Configure the ASA to Pass IPv6 Traffic

Document ID: 119012

Contributed by Kevin Klous, Cisco TAC Engineer.
Jun 29, 2015

Contents

Introduction

This document describes how to configure the Cisco Adaptive Security Appliance (ASA) in order to pass Internet Protocol Version 6 (IPv6) traffic in ASA Versions 7.0(1) and later.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

The information in this document is based on the Cisco ASA Versions 7.0(1) and later.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Background Information

Currently, IPv6 is still relatively new in terms of market penetration. However, IPv6 configuration assistance and troubleshooting requests have steadily increased. The purpose of this document is to address those needs and provide:

- A general overview of IPv6 usage
- The basic IPv6 configurations on the ASA
- Information about how to troubleshoot IPv6 connectivity through the ASA
- A list of the most common IPv6 problems and solutions, as identified by the Cisco Technical Assistance Center (TAC)

Note: Given that IPv6 is still in the early stages as an IPv4 replacement globally, this document will be periodically updated in order to maintain accuracy and relevance.

IPv6 Feature Information

Here is some important information about the IPv6 functionality:

- The IPv6 protocol was first introduced in ASA Version 7.0(1).
- Support for IPv6 in transparent mode was introduced in ASA Version 8.2(1).

IPv6 Overview

The IPv6 protocol was developed in the mid to late 1990s, primarily due to the fact that the public IPv4 address space moved quickly towards depletion. Although Network Address Translation (NAT) dramatically helped IPv4 and delayed this problem, it became undeniable that a replacement protocol would eventually be needed. The IPv6 protocol was officially detailed in RFC 2460 in December 1998. You can read more about the protocol in the official RFC 2460 document, located on the Internet Engineering Task Force (IETF) website.

IPv6 Improvements over IPv4

This section describes the improvements that are included with the IPv6 protocol versus the older IPv4 protocol.

Expanded Addressing Capabilities

The IPv6 protocol increases the IP address size from 32 bits to 128 bits in order to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto−configuration of addresses. The scalability of multicast routing is improved by the addition of a *scope* field to the multicast addresses. Additionally, a new type of address, called an *anycast address*, is defined. This is used in order to send a packet to any one node in a group.

Header Format Simplification

Some IPv4 header fields have been dropped or made optional in order to reduce the common−case processing cost of packet handling and in order to limit the bandwidth cost of the IPv6 header.

Improved Support for Extensions and Options

Changes in the way that the IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for the introduction of new options in the future.

Flow Labeling Capability

A new capability is added in order to enable the labeling of packets that belong to particular traffic *flows* for which the sender requests special handling, such as non−default Quality of Service (QoS) or *real−time* service.

Authentication and Privacy Capabilities

Extensions that are used in order to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

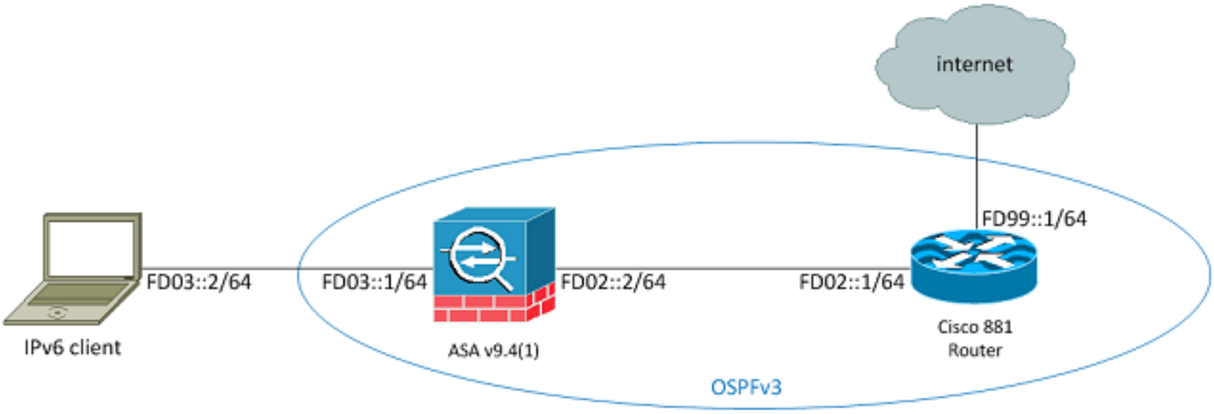
Configure

This section describes how to configure the Cisco ASA for the use of IPv6.

Note: Use the Command Lookup Tool (registered customers only) in order to obtain more information on the commands used in this section.

Network Diagram

This is the IPv6 topology for the examples that are used throughout this document:



Configure Interfaces for IPv6

In order to pass the IPv6 traffic through an ASA, you must first enable IPv6 on at least two interfaces. This example describes how to enable IPv6 in order to pass traffic from the inside interface on ***Gi0/0*** to the outside interface on ***Gi0/1***:

```
ASAv(config)# interface GigabitEthernet0/0
ASAv(config-if)# ipv6 enable

ASAv(config)# interface GigabitEthernet0/1
ASAv(config-if)# ipv6 enable
```

You can now configure the IPv6 addresses on both of the interfaces.

Note: In this example, the addresses in the Unique Local Addresses (ULA) space of fc00::/7 are used, so all of the addresses begin with ***FD*** (such as, fdxx:xxxx:xxxx....). Also, when you write IPv6 addresses, you can use double colons (***::***) in order to represent a line of zeros so that ***FD01::1/64*** is the same as ***FD01:0000:0000:0000:0000:0000:0000:00001***.

```
ASAv(config)# interface GigabitEthernet0/0
ASAv(config-if)# ipv6 address fd03::1/64
ASAv(config-if)# nameif inside
ASAv(config-if)# security-level 100

ASAv(config)# interface GigabitEthernet0/1
ASAv(config-if)# ipv6 address fd02::2/64
ASAv(config-if)# nameif outside
ASAv(config-if)# security-level 0
```

You should now have the basic Layer 2 (L2)/Layer 3 (L3) connectivity to an upstream router on the outside VLAN at address ***fd02::1***:

```
ASAv(config-if)# ping fd02::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to fd02::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/10 ms
```

Configure IPv6 Routing

Just as with IPv4, even though there is IPv6 connectivity with the hosts on the directly-connected subnet, you must still have the routes to the external networks in order to know how to reach them. The first example shows how to configure a static default route in order to reach all of the IPv6 networks via the outside interface with a next hop address of ***fd02::1***.

Configure Static Routing for IPv6

Use this information in order to configure static routing for IPv6:

```
ASAv(config)# ipv6 route outside 0::0/0 fd02::1
ASAv(config)# show ipv6 route

IPv6 Routing Table - 7 entries
Codes: C - Connected, L - Local, S - Static
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
            ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, B - BGP
L   fd02::2/128 [0/0]
     via ::, outside
C   fd02::/64 [0/0]
     via ::, outside
L   fd03::1/128 [0/0]
     via ::, inside
C   fd03::/64 [0/0]
     via ::, inside
L   fe80::/10 [0/0]
     via ::, inside
     via ::, outside
L   ff00::/8 [0/0]
     via ::, inside
     via ::, outside
S   ::/0 [1/0]
     via fd02::1, outsideASAv(config)#
```

As shown, there is now connectivity to a host on an external subnet:

```
ASAv(config)# ping fd99::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to fd99::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
ASAv(config)#
```

Note: If a dynamic routing protocol is desired in order to handle the routing for IPv6, then you can configure that as well. This is described in the next section.

Configure Dynamic Routing for IPv6 with OSPFv3

First, you should examine the Open Shortest Path First Version 3 (OSPFv3) configuration on the upstream Cisco 881 Series Integrated Services Router (ISR):

```
C881#show run | sec ipv6
ipv6 unicast-routing

!--- This enables IPv6 routing in the Cisco IOS®.

.....
 ipv6 ospf 1 area 0
 address-family ipv6 unicast
  passive-interface default
  no passive-interface Vlan302

!--- This is the interface to send OSPF Hellos to the ASA.

  default-information originate always
```

```
!--- Always distribute the default route.

  redistribute static
ipv6 route ::/0 FD99::2

!--- Creates a static default route for IPv6 to the internet.
```

Here is the relevant interface configuration:

```
C881#show run int Vlan302
interface Vlan302
....
 ipv6 address FD02::1/64
 ipv6 ospf 1 area 0
C881#
```

You can use ASA packet captures in order to verify that the OSPF *Hello* packets are seen from the ISR on the outside interface:

```
ASAv(config)# show run access-list test_ipv6
access-list test_ipv6 extended permit ip any6 any6
ASAv(config)# show cap
capture capout type raw-data access-list test_ipv6 interface outside
 [Capturing - 37976 bytes]
ASAv(config)# show cap capout

367 packets captured

   1: 11:12:04.949474       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
      neighbor sol: who has fe80::250:56ff:fe9d:34a8 [class 0xe0]
   2: 11:12:06.949444       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
      neighbor sol: who has fe80::250:56ff:fe9d:34a8 [class 0xe0]
   3: 11:12:07.854768       fe80::c671:feff:fe93:b516 > ff02::5: ip-proto-89 40
      [hlim 1]
   4: 11:12:07.946545       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
      neighbor sol: who has fe80::250:56ff:fe9d:34a8 [class 0xe0]
   5: 11:12:08.949459       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
      neighbor sol: who has fe80::250:56ff:fe9d:34a8 [class 0xe0]
   6: 11:12:09.542772       fe80::217:fff:fe17:af80 > ff02::5: ip-proto-89 40
      [hlim 1]
....
  13: 11:12:16.983011       fe80::c671:feff:fe93:b516 > ff02::5: ip-proto-89 40
      [hlim 1]
  14: 11:12:18.947170       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
      neighbor sol: who has fe80::250:56ff:fe9d:34a8 [class 0xe0]
  15: 11:12:19.394831       fe80::217:fff:fe17:af80 > ff02::5: ip-proto-89 40
      [hlim 1]
  16: 11:12:19.949444       fe80::250:56ff:fe9d:34a8 > ff02::1:ff9d:34a8: icmp6:
  21: 11:12:26.107477       fe80::c671:feff:fe93:b516 > ff02::5: ip-proto-89 40
      [hlim 1]
ASAv(config)#
```

In the previous packet capture, you can see that the OSPF (***ip−proto−89***) packets arrive from the IPv6 link−local address, which corresponds to the correct interface on the ISR:

```
C881#show ipv6 interface brief
......
Vlan302                [up/up]
    FE80::C671:FEFF:FE93:B516
    FD02::1
C881#
```

You can now create an OSPFv3 process on the ASA in order to establish an adjacency with the ISR:

```
ASAv(config)# ipv6 router ospf 1
ASAv(config-rtr)#  passive-interface default
ASAv(config-rtr)#  no passive-interface outside
ASAv(config-rtr)#  log-adjacency-changes
ASAv(config-rtr)#  redistribute connected
ASAv(config-rtr)#  exit
```

Apply the OSPF configuration to the ASA outside interface:

```
ASAv(config)# interface GigabitEthernet0/1
ASAv(config-if)#  ipv6 ospf 1 area 0
ASAv(config-if)#  end
```

This should cause the ASA to send the broadcast OSPF Hello packets on the IPv6 subnet. Enter the ***show ipv6 ospf neighbor*** command in order to verify adjacency with the router:

```
ASAv# show ipv6 ospf neighbor

Neighbor ID     Pri   State           Dead Time   Interface ID    Interface
    14.38.104.1 1     FULL/BDR        0:00:33     14              outside
```

You can also confirm the neighbor ID on the ISR, as it uses the highest configured IPv4 address for the ID by default:

```
C881#show ipv6 ospf 1
 Routing Process "ospfv3 1" with ID 14.38.104.1
 Supports NSSA (compatible with RFC 3101)
 Event-log enabled, Maximum number of events: 1000, Mode: cyclic
 It is an autonomous system boundary router
 Redistributing External Routes from,
    static
 Originate Default Route with always

!--- Notice the other OSPF settings that were configured.

 Router is not originating router-LSAs with maximum metric
....

C881#
```

The ASA should now have learned the default IPv6 route from the ISR. In order to confirm this, enter the ***show ipv6 route*** command:

```
ASAv# show ipv6 route

IPv6 Routing Table - 8 entries
Codes: C - Connected, L - Local, S - Static
       O - OSPF intra, OI - OSPF inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
           ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, B - BGP
O   2001:aaaa:aaaa:aaaa::/64 [110/10]
    via ::, outside
L   fd02::2/128 [0/0]
    via ::, outside
C   fd02::/64 [0/0]
    via ::, outside
L   fd03::1/128 [0/0]
    via ::, inside
C   fd03::/64 [0/0]
    via ::, inside
L   fe80::/10 [0/0]
    via ::, inside
    via ::, outside
L   ff00::/8 [0/0]
    via ::, inside
    via ::, outside
OE2  ::/0 [110/1], tag 1

!--- Here is the learned default route.

    via fe80::c671:feff:fe93:b516, outside
ASAv#
```

The basic configuration of the interface settings and routing features for IPv6 on the ASA is now complete.

Verify

There is currently no verification procedure available for this configuration.

Troubleshoot

The troubleshooting procedures for IPv6 connectivity follows most of the same methodology that is used in order to troubleshoot IPv4 connectivity, with a few differences. From a troubleshooting perspective, one of the most important differences between IPv4 and IPv6 is that the Address Resolution Protocol (ARP) no longer exists in IPv6. Instead of the use of ARP in order to resolve IP addresses on the local LAN segment, IPv6 uses a protocol called Neighbor Discovery (ND).

It is also important to understand that ND leverages Internet Control Message Protocol Version 6 (ICMPv6) for Media Access Control (MAC) address resolution. More information about IPv6 ND can be found in the ASA IPv6 Configuration guide in the IPv6 Neighbor Discovery section of the *CLI Book 1: Cisco ASA Series General Operations CLI Configuration Guide, 9.4* or in RFC 4861.

Currently, most IPv6−related troubleshooting involves either ND, routing, or subnet configuration problems. This is likely due to the fact that these are also the key differences between IPv4 and IPv6. The ND works differently than ARP, and internal network addressing is also quite different, as the use of NAT is highly discouraged in IPv6 and private addressing is no longer leveraged the way that it was in IPv4 (after RFC 1918). Once these differences are understood and/or the L2/L3 problems are resolved, the troubleshooting process at Layer 4 (L4) and above is essentially the same as that used for IPv4 because the TCP/UDP and higher−layer protocols function essentially the same (regardless of the IP version that is used).

Troubleshoot L2 Connectivity (ND)

The most basic command that is used in order to troubleshoot L2 connectivity with IPv6 is the ***show ipv6 neighbor [nameif]*** command, which is the equivalent of the ***show arp*** for IPv4.

Here is an example output:

```
ASAv(config)# show ipv6 neighbor outside
IPv6 Address                              Age Link−layer Addr State Interface
fd02::1                                     0 c471.fe93.b516  REACH   outside
fe80::c671:feff:fe93:b516                  32 c471.fe93.b516  DELAY   outside
fe80::e25f:b9ff:fe3f:1bbf                 101 e05f.b93f.1bbf  STALE   outside
fe80::b2aa:77ff:fe7c:8412                 101 b0aa.777c.8412  STALE   outside
fe80::213:c4ff:fe80:5f53                  101 0013.c480.5f53  STALE   outside
fe80::a64c:11ff:fe2a:60f4                 101 a44c.112a.60f4  STALE   outside
fe80::217:fff:fe17:af80                    99 0017.0f17.af80  STALE   outside
ASAv(config)#
```

In this output, you can see the successful resolution for the IPv6 address of ***fd02::1***, which belongs to the device with a MAC address of ***c471.fe93.b516***.

Note: You might notice that the same router interface MAC address appears twice in the previous output because the router also has a self−assigned link−local address for this interface. The link−local address is a device−specific address that can only be used for communication on the directly−connected network. Routers do not forward packets via link−local addresses, but rather they are only for communication on the directly−connected network segment. Many IPv6 routing protocols (such as OSPFv3) utilize link−local addresses in order to share routing protocol information on the L2 segment.

In order to clear the ND cache, enter the ***clear ipv6 neighbors*** command. If the ND fails for a particular host, you can enter the ***debug ipv6 nd*** command, as well as perform packet captures and verify the syslogs, in order to determine that which occurs at the L2 level. Remember that the IPv6 ND uses ICMPv6 messages in order to resolve the MAC addresses for IPv6 addresses.

IPv4 ARP Versus IPv6 ND

Consider this comparison table of ARP for IPv4 and ND for IPv6:

IPv4 ARP	IPv6 ND
ARP REQUEST (Who has 10.10.10.1?)	Neighbor Solicitation
ARP REPLY (10.10.10.1 is at dead.dead.dead)	Neighbor Advertisement

In the next scenario, the ND fails to resolve the MAC address of the *fd02::1* host that is located on the outside interface.

ND Debugs

Here is the output of ***debug ipv6 nd*** command:

```
ICMPv6-ND: Sending NS for fd02::1 on outside

!--- "Who has fd02::1"

ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: INCMP deleted: fd02::1
ICMPv6-ND: INCMP -> DELETE: fd02::1
ICMPv6-ND: DELETE -> INCMP: fd02::1
ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: Sending NA for fd02::2 on outside

!--- "fd02::2 is at dead.dead.dead"

ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: INCMP deleted: fd02::1
ICMPv6-ND: INCMP -> DELETE: fd02::1
ICMPv6-ND: DELETE -> INCMP: fd02::1

!--- Here is where the ND times out.

ICMPv6-ND: Sending NS for fd02::1 on outside
ICMPv6-ND: Sending NS for fd02::1 on outside
```

In this debug output, it *appears* that the Neighbor Advertisements from ***fd02::2*** are never received. You can check the packet captures in order to confirm whether this is actually the case.

ND Packet Captures

Note: As of ASA Release 9.4(1), access−lists are still required for IPv6 packet captures. An enhancement request has been filed in order to track this with Cisco bug ID CSCtn09836.

Configure the Access Control List (ACL) and packet captures:

```
ASAv(config)# access−list test_ipv6 extended permit ip any6 any6
ASAv(config)# cap capout interface outside access−list test_ipv6
```

Initiate a ping to ***fd02::1*** from the ASA:

```
ASAv(config)# show cap capout
....
23: 10:55:10.275284       fd02::2 > ff02::1:ff00:1: icmp6: neighbor sol: who has
 fd02::1 [class 0xe0]
  24: 10:55:10.277588       fd02::1 > fd02::2: icmp6: neighbor adv: tgt is fd02::1
   [class 0xe0]
  26: 10:55:11.287735       fd02::2 > ff02::1:ff00:1: icmp6: neighbor sol: who has
   fd02::1 [class 0xe0]
  27: 10:55:11.289642       fd02::1 > fd02::2: icmp6: neighbor adv: tgt is fd02::1
   [class 0xe0]
  28: 10:55:12.293365       fd02::2 > ff02::1:ff00:1: icmp6: neighbor sol: who has
   fd02::1 [class 0xe0]
  29: 10:55:12.298538       fd02::1 > fd02::2: icmp6: neighbor adv: tgt is fd02::1
   [class 0xe0]
  32: 10:55:14.283341       fd02::2 > ff02::1:ff00:1: icmp6: neighbor sol: who has
   fd02::1 [class 0xe0]
  33: 10:55:14.285690       fd02::1 > fd02::2: icmp6: neighbor adv: tgt is fd02::1
   [class 0xe0]
  35: 10:55:15.287872       fd02::2 > ff02::1:ff00:1: icmp6: neighbor sol: who has
   fd02::1 [class 0xe0]
  36: 10:55:15.289825       fd02::1 > fd02::2: icmp6: neighbor adv: tgt is fd02::1
   [class 0xe0]
```

As shown in the packet captures, the Neighbor Advertisements from ***fd02::1*** are received. However, the advertisements are not processed for some reason, as shown in the debug outputs. For further examination, you can view the syslogs.

ND Syslogs

Here are some example ND syslogs:

```
May 13 2015 10:55:10: %ASA−7−609001: Built local−host identity:fd02::2
May 13 2015 10:55:10: %ASA−6−302020: Built outbound ICMP connection for faddr
 ff02::1:ff00:1/0 gaddr fd02::2/0 laddr fd02::2/0(any)
May 13 2015 10:55:10: %ASA−3−325003: EUI−64 source address check failed.  Dropped
 packet from outside:fd02::1/0 to fd02::2/0 with source MAC address c471.fe93.b516.
May 13 2015 10:55:10: %ASA−3−313008: Denied IPv6−ICMP type=136, code=0 from fd02::1
 on interface outside
May 13 2015 10:55:11: %ASA−3−325003: EUI−64 source address check failed.  Dropped
 packet from outside:fd02::1/0 to fd02::2/0 with source MAC address c471.fe93.b516.
May 13 2015 10:55:11: %ASA−3−313008: Denied IPv6−ICMP type=136, code=0 from fd02::1
 on interface outside
May 13 2015 10:55:12: %ASA−3−325003: EUI−64 source address check failed.  Dropped
 packet from outside:fd02::1/0 to fd02::2/0 with source MAC address c471.fe93.b516.
May 13 2015 10:55:12: %ASA−3−313008: Denied IPv6−ICMP type=136, code=0 from fd02::1
 on interface outside
May 13 2015 10:55:14: %ASA−3−325003: EUI−64 source address check failed.  Dropped
 packet from outside:fd02::1/0 to fd02::2/0 with source MAC address c471.fe93.b516.
May 13 2015 10:55:14: %ASA−3−313008: Denied IPv6−ICMP type=136, code=0 from fd02::1
 on interface outside
```

```
May 13 2015 10:55:15: %ASA-3-325003: EUI-64 source address check failed.  Dropped
 packet from outside:fd02::1/0 to fd02::2/0 with source MAC address c471.fe93.b516.
May 13 2015 10:55:15: %ASA-3-313008: Denied IPv6-ICMP type=136, code=0 from fd02::1
 on interface outside
```

Within these syslogs, you can see that the ND Neighbor Advertisement packets from the ISR at ***fd02::1*** are dropped due to failed Modified Extended Unique Identifier (EUI) 64 (Modified EUI−64) format checks.

Tip: Refer to the *Modified EUI−64 Address Encoding* section of this document for more information about this specific problem. This troubleshooting logic can be applied to all kinds of drop reasons as well, such as when the ACLs do not permit ICMPv6 on a specific interface or when Unicast Reverse Path Forwarding (uRPF) check failures occur, both of which can cause L2 connectivity issues with IPv6.

Troubleshoot Basic IPv6 Routing

The troubleshooting procedures for routing protocols when IPv6 is used are essentially the same as those when IPv4 is used. The use of ***debug*** and ***show*** commands, as well as packet captures, are useful with attempts to ascertain the reason that a routing protocol does not behave as expected.

Routing Protocol Debugs for IPv6

This section provides the useful debug commands for IPv6.

Global IPv6 Routing Debugs

You can use the ***debug ipv6 routing*** debug in order to troubleshoot all of the IPv6 routing table changes:

```
ASAv# clear ipv6 ospf 1 proc

Reset OSPF process? [no]: yes
ASAv# IPv6RT0: ospfv3 1, Route update to STANDBY with epoch: 2 for
 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ospfv3 1, Delete 2001:aaaa:aaaa:aaaa::/64 from table
IPv6RT0: ospfv3 1, Delete backup for fd02::/64
IPv6RT0: ospfv3 1, Route update to STANDBY with epoch: 2 for ::/0
IPv6RT0: ospfv3 1, Delete ::/0 from table
IPv6RT0: ospfv3 1, ipv6_route_add_core for 2001:aaaa:aaaa:aaaa::/64 [110/10],
 next-hop :: nh_source :: via interface outside route-type 2
IPv6RT0: ospfv3 1, Add 2001:aaaa:aaaa:aaaa::/64 to table
IPv6RT0: ospfv3 1, Added next-hop :: over outside for 2001:aaaa:aaaa:aaaa::/64,
 [110/10]
IPv6RT0: ospfv3 1, ipv6_route_add_core Route update to STANDBY with epoch: 2 for
 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ipv6_route_add_core: input add 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ipv6_route_add_core: output add 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ospfv3 1, ipv6_route_add_core for fd02::/64 [110/10], next-hop ::
 nh_source :: via interface outside route-type 2
IPv6RT0: ospfv3 1, ipv6_route_add_core for ::/0 [110/1], next-hop
 fe80::c671:feff:fe93:b516
 nh_source fe80::c671:feff:fe93:b516 via interface outside route-type 16
IPv6RT0: ospfv3 1, Add ::/0 to table
IPv6RT0: ospfv3 1, Added next-hop fe80::c671:feff:fe93:b516 over outside for ::/0,
 [110/1]
IPv6RT0: ospfv3 1, ipv6_route_add_core Route update to STANDBY with epoch: 2 for ::/0
IPv6RT0: ipv6_route_add_core: input add ::/0
IPv6RT0: ipv6_route_add_core: output add ::/0
IPv6RT0: ospfv3 1, ipv6_route_add_core for 2001:aaaa:aaaa:aaaa::/64 [110/10],
 next-hop :: nh_source :: via interface outside route-type 2
IPv6RT0: ospfv3 1, Route add 2001:aaaa:aaaa:aaaa::/64 [owner]
IPv6RT0: ospfv3 1, ipv6_route_add_core Route update to STANDBY with epoch: 2 for
 2001:aaaa:aaaa:aaaa::/64
```

```
IPv6RT0: ipv6_route_add_core: input add 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ipv6_route_add_core: output add 2001:aaaa:aaaa:aaaa::/64
IPv6RT0: ospfv3 1, ipv6_route_add_core for fd02::/64 [110/10], next-hop ::
 nh_source :: via interface outside route-type 2
IPv6RT0: ospfv3 1, Reuse backup for fd02::/64, distance 110
IPv6RT0: ospfv3 1, ipv6_route_add_core for ::/0 [110/1], next-hop
 fe80::c671:feff:fe93:b516 nh_source fe80::c671:feff:fe93:b516 via interface outside
 route-type 16
IPv6RT0: ospfv3 1, Route add ::/0 [owner]
IPv6RT0: ospfv3 1, ipv6_route_add_core Route update to STANDBY with epoch: 2 for ::/0
IPv6RT0: ipv6_route_add_core: input add ::/0
IPv6RT0: ipv6_route_add_core: output add ::/0
```

OSPFv3 Debugs

You can use the ***debug ipv6 ospf*** command in order to troubleshoot OSPFv3 issues:

```
ASAv# debug ipv6 ospf ?

  adj                 OSPF adjacency events
  database-timer      OSPF database timer
  events              OSPF events
  flood               OSPF flooding
  graceful-restart    OSPF Graceful Restart processing
  hello               OSPF hello events
  ipsec               OSPF ipsec events
  lsa-generation      OSPF lsa generation
  lsdb                OSPF database modifications
  packet              OSPF packets
  retransmission      OSPF retransmission events
  spf                 OSPF spf
```

Here is an example output for all of the debugs that are enabled after the OSPFv3 process is restarted:

```
ASAv# clear ipv6 ospf 1
OSPFv3: rcv. v:3 t:1 l:44 rid:192.168.128.115
      aid:0.0.0.0 chk:a9ac inst:0 from outside
OSPFv3: Rcv hello from 192.168.128.115 area 0 from outside fe80::217:fff:fe17:af80
 interface ID 142
OSPFv3: End of hello processingpr
OSPFv3: rcv. v:3 t:1 l:44 rid:14.38.104.1
      aid:0.0.0.0 chk:bbf3 inst:0 from outside
OSPFv3: Rcv hello from 14.38.104.1 area 0 from outside fe80::c671:feff:fe93:b516
 interface ID 14
OSPFv3: End of hello processingo
ASAv# clear ipv6 ospf 1 process

Reset OSPF process? [no]: yes
ASAv#
OSPFv3: Flushing External Links
  Insert LSA 0 adv_rtr 172.16.118.1, type 0x4005 in maxage
OSPFv3: Add Type 0x4005 LSA ID 0.0.0.0 Adv rtr 172.16.118.1 Seq 80000029 to outside
 14.38.104.1 retransmission list
....

!--- The neighbor goes down:

OSPFv3: Neighbor change Event on interface outside
OSPFv3: DR/BDR election on outside
OSPFv3: Elect BDR 14.38.104.1
OSPFv3: Elect DR 192.168.128.115
OSPFv3: Schedule Router LSA area: 0, flag: Change
OSPFv3: Schedule Router LSA area: 0, flag: Change
OSPFv3: Schedule Prefix DR LSA intf outside
OSPFv3: Schedule Prefix Stub LSA area 0
```

```
OSPFv3: 14.38.104.1 address fe80::c671:feff:fe93:b516 on outside is dead, state DOWN
....

!--- The neighbor resumes the exchange:

OSPFv3: Rcv DBD from 14.38.104.1 on outside seq 0xd09 opt 0x0013 flag 0x7 len 28
  mtu 1500 state EXSTART
OSPFv3: First DBD and we are not SLAVE
OSPFv3: rcv. v:3 t:2 l:168 rid:14.38.104.1
      aid:0.0.0.0 chk:5aa3 inst:0 from outside
OSPFv3: Rcv DBD from 14.38.104.1 on outside seq 0x914 opt 0x0013 flag 0x2 len 168
  mtu 1500 state EXSTART
OSPFv3: NBR Negotiation Done. We are the MASTER
OSPFv3: outside Nbr 14.38.104.1: Summary list built, size 0
OSPFv3: Send DBD to 14.38.104.1 on outside seq 0x915 opt 0x0013 flag 0x1 len 28
OSPFv3: rcv. v:3 t:2 l:28 rid:192.168.128.115
      aid:0.0.0.0 chk:295c inst:0 from outside
OSPFv3: Rcv DBD from 192.168.128.115 on outside seq 0xfeb opt 0x0013 flag 0x7 len 28
  mtu 1500 state EXSTART
OSPFv3: NBR Negotiation Done. We are the SLAVE
OSPFv3: outside Nbr 192.168.128.115: Summary list built, size 0
OSPFv3: Send DBD to 192.168.128.115 on outside seq 0xfeb opt 0x0013 flag 0x0 len 28
OSPFv3: rcv. v:3 t:2 l:28 rid:14.38.104.1
      aid:0.0.0.0 chk:8d74 inst:0 from outside
OSPFv3: Rcv DBD from 14.38.104.1 on outside seq 0x915 opt 0x0013 flag 0x0 len 28
  mtu 1500 state EXCHANGE
....

!--- The routing is re-added to the OSPFv3 neighbor list:

OSPFv3: Add Router 14.38.104.1 via fe80::c671:feff:fe93:b516, metric: 10
  Router LSA 14.38.104.1/0, 1 links
    Link 0, int 14, nbr 192.168.128.115, nbr int 142, type 2, cost 1
      Ignore newdist 11 olddist 10
```

Enhanced Interior Gateway Routing Protocol (EIGRP)

The EIGRP on the ASA does not support the use of IPv6. Refer to the Guidelines for EIGRP section of the *CLI Book 1: Cisco ASA Series General Operations CLI Configuration Guide, 9.4* for more information.

Border Gateway Protocol (BGP)

This ***debug*** command can be used in order to troubleshoot BGP when IPv6 is used:

```
ASAv# debug ip bgp ipv6 unicast  ?

  X:X:X:X::X  IPv6 BGP neighbor address
  keepalives  BGP keepalives
  updates     BGP updates
  <cr>
```

Useful Show Commands for IPv6

You can use these ***show*** commands in order to troubleshoot IPv6 issues:

- ***show ipv6 route***
- ***show ipv6 interface brief***
- ***show ipv6 ospf <process ID>***

- ***show ipv6 traffic***
- ***show ipv6 neighbor***
- ***show ipv6 icmp***

Packet Tracers with IPv6

You can use the built−in packet tracer functionality with IPv6 on the ASA in the same way as with IPv4. Here is an example where the packet−tracer functionality is used in order to simulate the inside host at ***fd03::2***, which attempts to connect to a web server at ***5555::1*** that is located on the Internet with the default route that is learned from the ***881*** interface via OSPF:

```
ASAv# packet−tracer input inside tcp fd03::2 10000 5555::1 80 detailed

Phase: 1
Type: ACCESS−LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
 Forward Flow based lookup yields rule:
 in  id=0x7fffd59ca0f0, priority=1, domain=permit, deny=false
        hits=2734, user_data=0x0, cs_id=0x0, l3_type=0xdd86
        src mac=0000.0000.0000, mask=0000.0000.0000
        dst mac=0000.0000.0000, mask=0100.0000.0000
        input_ifc=inside, output_ifc=any

Phase: 2
Type: ROUTE−LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next−hop fe80::c671:feff:fe93:b516 using egress ifc  outside

Phase: 3
Type: NAT
Subtype: per−session
Result: ALLOW
Config:
Additional Information:
 Forward Flow based lookup yields rule:
 in  id=0x7fffd589cc30, priority=1, domain=nat−per−session, deny=true
        hits=1166, user_data=0x0, cs_id=0x0, reverse, use_real_addr, flags=0x0,
         protocol=6
        src ip/id=::/0, port=0, tag=any
        dst ip/id=::/0, port=0, tag=any
        input_ifc=any, output_ifc=any

<<truncated output>>

Result:
input−interface: inside
input−status: up
input−line−status: up
output−interface: outside
output−status: up
output−line−status: up
Action: allow

ASAv#
```

Notice that the egress MAC address is the link−local address of the 881 interface. As mentioned previously, for many dynamic routing protocols, routers use link−local IPv6 addresses in order to establish adjacencies.

Complete List of IPv6−Related ASA Debugs

Here are the debugs that can be used in order to troubleshoot IPv6 issues:

```
ASAv# debug ipv6 ?

  dhcp       IPv6 generic dhcp protocol debugging
  dhcprelay  IPv6 dhcp relay debugging
  icmp       ICMPv6 debugging
  interface  IPv6 interface debugging
  mld        IPv6 Multicast Listener Discovery debugging
  nd         IPv6 Neighbor Discovery debugging
  ospf       OSPF information
  packet     IPv6 packet debugging
  routing    IPv6 routing table debugging
```

Common IPv6−Related Problems

This section describes how to troubleshoot the most common IPv6−related issues.

Improperly Configured Subnets

Many IPv6 TAC cases are generated due to a general lack of knowledge about how IPv6 functions, or due to administrator attempts to implement IPv6 with the use of IPv4−specific processes.

For example, the TAC has seen cases where an administrator has been assigned a \56 block of IPv6 addresses by an Internet Service Provider (ISP). The administrator then assigns an address and the full \56 subnet to the ASA outside interface and chooses some internal range to use for the inside servers. However, with IPv6, all of the internal hosts should also use routable IPv6 addresses, and the IPv6 address block should be broken down into smaller subnets as needed. In this scenario, you can create many \64 subnets as a part of the \56 block that has been allocated.

Tip: Refer to RFC 4291 for additional information.

Modified EUI 64 Encoding

The ASA can be configured in order to require modified EUI−64−encoded IPv6 addresses. The EUI, as per RFC 4291, allows a host to assign itself a unique 64−Bit IPv6 interface identifier (EUI−64). This feature is an advantage over IPv4, as it removes the requirement to utilize DHCP for the IPv6 address assignment.

If the ASA is configured in order to require this enhancement via the ***ipv6 enforce−eui64 nameif*** command, then it will likely drop many Neighbor Discovery solicitations and Advertisements from other hosts on the local subnet.

Tip: For more information, refer to the Understanding IPv6 EUI−64 Bit Address Cisco Support Community document.

Clients Use Temporary IPv6 Addresses by Default

By default, many client Operating Systems (OSs), such as Microsoft Windows Versions 7 and 8, Macintosh OS−X, and Linux−based systems, use self−assigned *temporary* IPv6 addresses for extended privacy via IPv6 Stateless Address Autoconfiguration (SLAAC).

The Cisco TAC has seen some cases where this caused unexpected problems in environments because the hosts generate traffic from the temporary address and not the statically−assigned address. As a result, the ACLs and the host−based routes might cause the traffic to either become dropped or improperly routed, which causes the host communication to fail.

There are two methods that are used in order to address this situation. The behavior can be disabled individually on the client systems, or you can disable this behavior on the ASA and Cisco IOS® routers. On the ASA or router side, you must modify the Router Advertisement (RA) message flag that triggers this behavior.

Refer to the next sections in order to disable this behavior on the individual clients systems.

Microsoft Windows

Complete these steps in order to disable this behavior on Microsoft Windows systems:

1. In Microsoft Windows, open an Elevated Command Prompt (run as administrator).
2. Enter this command in order to disable the random IP address generation feature, and then press ***Enter***:

   ```
   netsh interface ipv6 set global randomizeidentifiers=disabled
   ```
3. Enter this command in order to force Microsoft Windows to use the EUI−64 standard:

   ```
   netsh interface ipv6 set privacy state=disabled
   ```
4. Reboot the machine in order to apply the changes.

Macintosh OS−X

In a terminal, enter this command in order to disable IPv6 SLAAC on the host until the next reboot:

```
sudo sysctl −w net.inet6.ip6.use_tempaddr=0
```

In order to make the configuration permanent, enter this command:

```
sudo sh −c 'echo net.inet6.ip6.use_tempaddr=0 >> /etc/sysctl.conf'
```

Linux

In a terminal shell, enter this command:

```
sysctl −w net.ipv6.conf.all.use_tempaddr=0
```

Disable SLAAC Globally from the ASA

The second method that is used in order to address this behavior is to modify the RA message that is sent from the ASA to the clients, which triggers the use of SLAAC. In order to modify the RA message, enter this command from *Interface Configuration* mode:

```
ASAv(config)# interface gigabitEthernet 1/1
ASAv(config−if)# ipv6 nd prefix 2001::db8/32 300 300 no−autoconfig
```

This command modifies the RA message that is sent by the ASA so that the A−bit flag is not set, and the clients do not generate a temporary IPv6 address.

Tip: Refer to RFC 4941 for additional information.

IPv6 FAQs

This section describes some frequently asked questions in regards to the use of IPv6.

Can I pass traffic for both IPv4 and IPv6 on the same interface, at the same time?

Yes. You must simply enable IPv6 on the interface and assign both an IPv4 and an IPv6 address to the interface, and it handles both types of traffic simultaneously.

Can I apply both IPv6 and IPv4 ACLs to the same interface?

You can do this in ASA versions earlier than Version 9.0(1). As of ASA Version 9.0(1), all ACLs on the ASA are *unified*, which means that an ACL supports a mix of both IPv4 and IPv6 entries in the same ACL.

In ASA Versions 9.0(1) and later, the ACLs are simply merged together and the single, unified ACL is applied to the interface via the ***access−group*** command.

Does the ASA support QoS for IPv6?

Yes. The ASA supports policing and priority queuing for IPv6 in the same way that it does with IPv4.

As of ASA Version 9.0(1), all ACLs on the ASA are *unified*, which means that an ACL supports a mix of both IPv4 and IPv6 entries in the same ACL. As a result, any QoS commands that are enacted on a class−map that matches an ACL take action on both the IPv4 and IPv6 traffic.

Should I use NAT with IPv6?

Although NAT can be configured for IPv6 on the ASA, the use of NAT in IPv6 is highly discouraged and unnecessary, given the near infinite amount of available, globally−routable IPv6 addresses.

If NAT is required in an IPv6 scenario, you can find more information about how to configure it in the IPv6 NAT Guidelines section of the *CLI Book 2: Cisco ASA Series Firewall CLI Configuration Guide, 9.4*.

Note: There are some guidelines and limitations that should be considered when you implement NAT with IPv6.

Why do I see the link−local IPv6 addresses in the *show failover* command output?

In IPv6, ND uses link−local addresses in order to perform L2 address resolution. For this reason, the IPv6 addresses for the monitored interfaces in the ***show failover*** command output show the link−local address and not the global IPv6 address that is configured on the interface. This is expected behavior.

Known Caveats/Enhancement Requests

Here are some known caveats in regards to the use of IPv6:

- Cisco bug ID CSCtn09836 *ASA 8.x capture "match" clause doesn't catch IPv6 traffic*
- Cisco bug ID CSCuq85949 *ENH: ASA IPv6 support for WCCP*
- Cisco bug ID CSCut78380 *ASA IPv6 ECMP routing does not load balance traffic*

Related Information

- ***RFC 2460 Internet Protocol, Version 6 (IPv6) Specification***
- ***RFC 4291 IP Version 6 Addressing Architecture***
- ***RFC 4861 Neighbor Discovery for IP Version 6 (IPv6)***
- ***CLI Book 1: Cisco ASA Series General Operations CLI Configuration Guide, 9.4 IPv6***
- ***AnyConnect SSL over IPv4+IPv6 to ASA Configuration***
- ***Technical Support & Documentation Cisco Systems***