

## Chapter 34

# Internet Protocol version 6 (IPv6)

Introduction .....	34-3
Overview of IPv6 .....	34-3
The 6bone .....	34-4
IPv6 Addresses and Prefixes .....	34-4
IPv6 Headers .....	34-5
The Internet Control Message Protocol (ICMPv6) .....	34-8
IPv6 Routing .....	34-11
IPv6 Filtering .....	34-12
Integration of IPv4 and IPv6 .....	34-12
IPv6 on the Switch .....	34-12
Enabling IPv6 .....	34-13
IPv6 Interfaces and Addresses .....	34-13
Extension Header Processing .....	34-15
Routing Table Processing and RIPv6 .....	34-15
Neighbour Discovery .....	34-16
IPv6 Filtering .....	34-17
IPv6 Fragmentation .....	34-18
Telnet v6 .....	34-18
Ping .....	34-19
Secure Shell .....	34-20
Tunnelling IPv6 Packets over IPv4 .....	34-20
Configuration Examples .....	34-23
Basic Routing .....	34-23
Dynamic Routing with RIPv6 .....	34-25
Dynamic (6-to-4) Tunnelling over an IPv4 Network .....	34-29
Static Tunnelling over an IPv4 Network .....	34-31
IPv6 Filters .....	34-33
Command Reference .....	34-35
add ipv6 6to4 .....	34-35
add ipv6 filter .....	34-36
add ipv6 host .....	34-41
add ipv6 interface .....	34-42
add ipv6 nd .....	34-45
add ipv6 prefix .....	34-46
add ipv6 rip .....	34-47
add ipv6 route .....	34-48
add ipv6 tunnel .....	34-49
create ipv6 interface .....	34-50
delete ipv6 6to4 .....	34-51
delete ipv6 filter .....	34-51
delete ipv6 host .....	34-52

delete ipv6 interface .....	34-52
delete ipv6 nd .....	34-54
delete ipv6 prefix .....	34-55
delete ipv6 rip .....	34-56
delete ipv6 route .....	34-57
delete ipv6 tunnel .....	34-58
destroy ipv6 interface .....	34-59
disable ipv6 .....	34-59
disable ipv6 advertise .....	34-60
disable ipv6 debug .....	34-60
disable ipv6 mtudiscovery .....	34-61
disable ipv6 rip .....	34-61
enable ipv6 .....	34-61
enable ipv6 advertise .....	34-62
enable ipv6 debug .....	34-62
enable ipv6 mtudiscovery .....	34-63
enable ipv6 rip .....	34-63
reset ipv6 ndcache .....	34-64
set ipv6 filter .....	34-65
set ipv6 interface .....	34-68
set ipv6 mtu .....	34-69
set ipv6 nd .....	34-70
set ipv6 prefix .....	34-72
set ipv6 route preference .....	34-73
show ipv6 .....	34-74
show ipv6 counter .....	34-76
show ipv6 filter .....	34-80
show ipv6 host .....	34-82
show ipv6 interface .....	34-83
show ipv6 multicast .....	34-85
show ipv6 ndcache .....	34-86
show ipv6 ndconfig .....	34-87
show ipv6 rip .....	34-89
show ipv6 route .....	34-91
show ipv6 route multicast .....	34-93
show ipv6 route preference .....	34-94
show ipv6 tunnel .....	34-95
show ipv6 udp .....	34-96

## Introduction

---

This chapter describes the main features of IPv6, the switch's implementation of IPv6 and how to configure and operate IPv6 on the switch.

This chapter describes the following IPv6 features:

- linking together networks that run IPv6.
- allowing address autoconfiguration of hosts connected to the switch.
- enabling IPv6 to operate over existing IPv4-based networks.

For more information about multicast support for IPv6 on the switch, see [Chapter 36, IPv6 Multicasting](#).

For more information about configuring security features for IPv6, see [Chapter 49, IP Security \(IPsec\)](#) and [Chapter 44, Compression and Encryption Services](#).

IPv6 is enabled with a special feature licence. To obtain a special feature licence contact an Allied Telesis authorised distributor or reseller.

## Overview of IPv6

---

IPv6 is the next generation of the Internet Protocol (IP). It has primarily been developed to solve the problem of the eventual exhaustion of the IPv4 address space, but also offers other enhancements. IPv6 addresses are 16 bytes long, in contrast to IPv4's 4 byte addresses. Other features of IPv6 include:

- Address structure improvements:
  - globally unique addresses with more levels of addressing hierarchy to reduce the size of routing tables
  - autoconfiguration of addresses by hosts
  - improved scalability of multicast routing by adding a "scope" field to multicast addresses
  - a new type of address, the "anycast address", which sends a packet to any one of a group of devices
- Removal of the need for packet fragmentation en-route by dynamic determination of the largest packet size that is supported by every link in the path. A link's MTU (Maximum Transmission Unit) must be at least 1280 bytes, compared with 576 bytes for IPv4.
- Traffic Class, which allows a packet to be labelled with an appropriate priority. If the network becomes congested, the lowest priority packets are dropped.
- Flow labels, which indicate to intermediate routers that packets are part of a flow, and that this flow requires a particular type of service. This feature enables, for example, real-time processing of data streams. It also increases routing speed because the forwarding router need only check the flow label, not the rest of the header. The handling indicated by the flow label can be done by the IPv6 Hop-by-Hop header, or by a separate protocol such as RSVP.
- Mandatory authentication and data integrity protocols through IPsec. IPsec is optional in IPv4.

An IPv6 network can contain three types of nodes, where *node* is a general term that refers to any IPv6-aware device. A *host* is a device on the network that is not a router. For example, a host may be a printer or a computer. A router may also act as a host. A *router* is a device on the network that directs the flow of IPv6 packets. For example, a router may be a router or a Layer 3 switch. A *destination* is a host to which packets are specifically sent.

## The 6bone

The 6bone is an experimental virtual network of nodes that support IPv6 packets, tunnelled together through the existing IPv4 Internet. Most of the nodes are workstations or similar machines, with IPv6-capable operating systems. The theory of tunnelling IPv6 packets over an IPv4 network is outlined in [“Integration of IPv4 and IPv6” on page 34-12](#).

The 6bone is part of the transition to IPv6. Its purpose is to provide an environment in which IPv6 can be tested and procedures for IPv6 can be developed. When IPv6 is sufficiently developed and being used widely, the 6bone will probably disappear.

## IPv6 Addresses and Prefixes

IPv6 addresses are hexadecimal, and are made up of eight pairs of octets separated by colons. An example of a valid address is fe80:0000:0000:0260:0000:97ff:64aa. In the interests of brevity, addresses can be abbreviated in two ways:

- Leading zeros can be omitted, so this address can be written as fe80:0:0:0:260:0:97ff:64aa.
- Consecutive zeros can be replaced with a double colon, so this address can be written as fe80::260:0:97ff:64aa. Note that a double colon can replace any number of consecutive zeros, but an address can contain only one double colon.

Like IPv4 addresses, a proportion of the leftmost bits of the IPv6 address can be used to indicate the subnet, rather than a single node. This part of the address is called the *prefix*. Prefixes provide the equivalent functionality to a subnet mask in IPv4, allowing a subnet to be addressed, rather than a single node. If a prefix is specified, the IPv6 address is followed by a slash and the number of bits that represent the prefix. For example, 3ffe::/16 indicates that the first 16 bits (3ffe) of the address 3ffe:0:0:0:0:0:0:0 represent the prefix.

Like IPv4 addresses, IPv6 addresses are attached to interfaces.

### Unicast Addresses

A unicast address is attached to a single interface and delivers packets only to that interface.

The following special addresses have been defined:

- IPv4-compatible and IPv4-mapped addresses. IPv4-compatible addresses are used to tunnel IPv6 packets across an IPv4 network. IPv4-mapped addresses are used by an IPv6 host to communicate with an IPv4 host. The IPv6 host addresses the packet to the mapped address.
- Link-local addresses, which can be used on the local network that the interface is attached to. The link-local prefix is fe80::/10. Different interfaces on a device may have the same link-local address.

- Site-local addresses, which are used in internal or private networks. These addresses are analogous to the IPv4 private addresses 10.x.x.x and 192.168.x.x.
- The Loopback address, consisting of ::1, which is the equivalent of the IPv4 loopback address, and allows a host to send packets to itself.
- The Unspecified address, consisting of ::, which is the equivalent of the IPv4 unspecified address, and is used as a source address by hosts during the autoconfiguration process.

**Multicast addresses** For information about IPv6 multicast group management and multicast routing, see [Chapter 36, IPv6 Multicasting](#).

IPv6 multicast addresses also provide the equivalent functionality to broadcast addresses in IPv4. Broadcast addresses are not supported in IPv6. A multicast address identifies a group of interfaces, and packets are sent to all interfaces in that group.

Among the special addresses that have been defined are addresses that allow multicasting to:

- All interfaces on a particular host (FF01::1)
- All nodes on a local network (FF01::2)
- All routers on the local link (FF02::2)
- All routers on the local site (FF05::2).

**Anycast addresses** An *anycast* address is a unicast address that is attached to more than one interface. If a packet is sent to an anycast address it is delivered to the nearest interface with that address, with the definition of “nearest” depending on the protocol used for routing. If the protocol is RIPv6, the nearest interface is the one that is the shortest number of hops away.

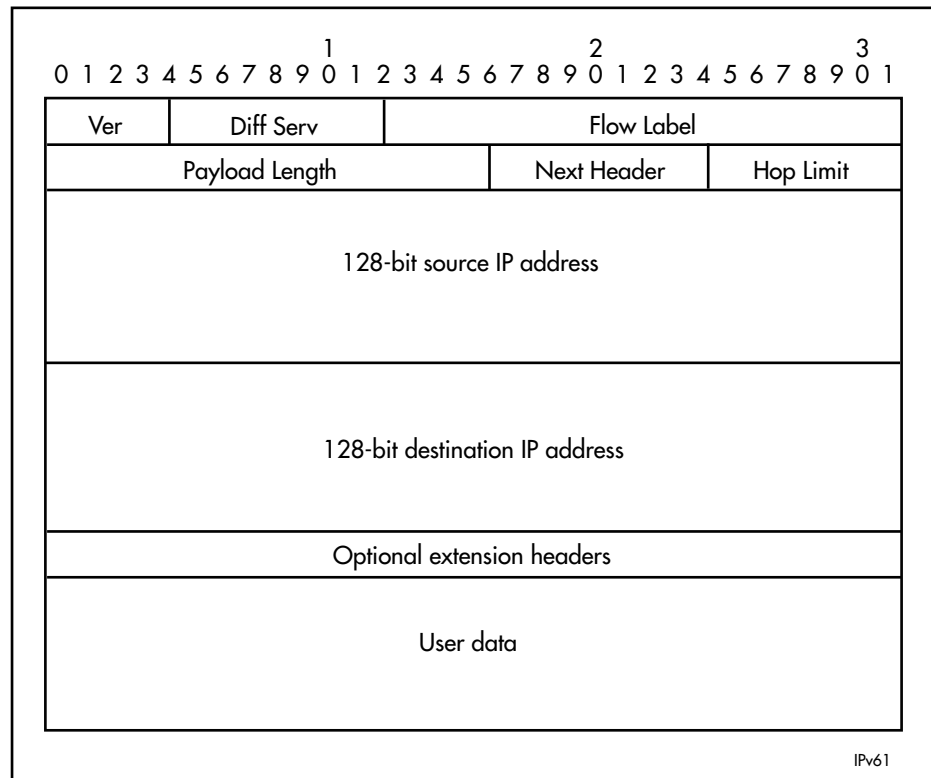
Anycast addresses can be assigned to routers only, and packets cannot originate from an anycast address. A router must be configured to know that it is using an anycast address because the address format cannot be distinguished from that of a unicast address.

Only one anycast address has been predefined: the subnet-router address. The subnet-router address sends messages to the nearest router on a subnet and consists of the subnet’s prefix followed by zeros.

## IPv6 Headers

The basic unit of data sent through an internet is called a *packet* in IPv6. A packet consists of a *header* followed by the *data*. The following figure shows the IPv6 packet.

Figure 34-1: IPv6 packet



The following table describes fields in an IPv6 packet.

Field	Function
Ver	Version of the IP protocol that created the packet. For IPv6, this field has a value of 6.
Differentiated Services	8-bit value that contains the 6-bit DSCP and is used to sort traffic as part of a Quality of Service system. For more information, see RFC 2474, <i>Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers</i> .
Flow Label	20-bit value that indicates the data flow to which this packet belongs. This flow may be handled in a particular way.
Payload Length	Length of the user data portion of the packet. If the data payload is larger than 64 kB, the length is given in the optional "Jumbo Payload" header and the Payload Length header is given a value of zero.
Next Header	Number that indicates the type of header that immediately follows the basic IP header. This header type may be an optional IPv6 extension header, a relevant IPv4 option header, or another protocol, such as TCP or ICMPv6. The IPv6 extension header values are: 0 (Hop-by-Hop Options Header) 43 (IPv6 Routing Header) 44 (IPv6 Fragment Header) 50 (Encapsulating Security Payload) 51 (IPv6 Authentication Header) 59 (No Next Header) 60 (Destination Options Header)

Field	Function
Hop Limit	Field that is the equivalent of the IPv4 Time To Live field, measured in hops.
Source IP address	128-bit IPv6 address of the sender.
Destination IP address	128-bit IPv6 address of the recipient.
Optional extension headers	Headers for less-frequently used information.
User data	Payload.

### Basic IPv6 header

The header contains information necessary to move the packet across the internet. It must be able to cope with missing and duplicated packets as well as possible fragmentation (and reassembly) of the original packet.

The IPv6 header is similar to the shorter IPv4 header, which is described in [Figure 23-1](#) and [Table 23-1](#) on page 23-6 of [Chapter 23, Internet Protocol \(IP\)](#). The IPv6 header is twice as long as the IPv4 header (40 bytes instead of 20 bytes) and contains four times the address space size (128 bits instead of 32 bits).

The IPv6 header no longer contains the header length, identification, flags, fragment offset, and header checksum fields. Some of these options are placed in extension headers. The Time To Live field is replaced with a hop limit, and the IPv4 Type of Service field is replaced with a Differentiated Services field. The Differentiated Services field contains the DSCP bits, used in a Quality of Service (QoS) regime. The following table explains IPv4 header fields that changed in IPv6.

Changed Field	Description
Type of Service	The type of service that a connection should receive is indicated in IPv6 by the Flow Label field in the IPv6 header.
Fragmentation information (the Identification field, the Flags field and the Fragment Offset field)	In most cases fragmentation does not occur in IPv6. If it does, packets are fragmented at their source and not en route. Therefore, the fragmentation information is contained in an extension header to reduce the size of the basic IPv6 header.
Header Checksum	This option has not been provided in IPv6. This is because transport protocols implement checksums and because of the availability of the IPsec authentication header (AH) in IPv6.
Options	Extension headers handle all the optional values associated with IPv6 packets. The biggest advantage of this scheme is that the size of the basic IP header is a constant.

### Extension headers

In IPv6 many of the less commonly used fields in the IPv4 header (or their equivalents) have become extension headers, which are placed after the basic IPv6 header. The length of each header must be a multiple of 8 bytes.

The first extension header is identified by the Next Header field in the basic IPv6 header. Any subsequent extension headers are identified by an 8-bit "Next Header" value at the beginning of the preceding extension header.

IPv6 nodes that originate packets are required to place extension headers in a specific order:

1. The basic IPv6 header, which must come immediately before the extension headers.
2. The Hop-by-Hop header, which specifies options that must be examined by every node in the routing path.
3. A Destination Options header, to specify options to be processed by the first destination or final destination. The destination options header is the only extension header that may be present more than once in the IPv6 packet.
4. The Routing header, which allows a static path to be specified for the packet, if the dynamically-determined path is undesirable.
5. The Fragment header, which indicates that the source node has fragmented the packet, and contains information about the fragmentation.
6. The Authentication header (AH), which verifies the integrity of the packet and its headers. The AH is an IPsec feature, and is described in [“IP Security \(IPsec\)” on page 49-4 of Chapter 49, IP Security \(IPsec\)](#).
7. The Encapsulating Security Payload (ESP) header, which encrypts a packet and verifies the integrity of its contents. The ESP is an IPsec feature, and is described in [“IP Security \(IPsec\)” on page 49-4 of Chapter 49, IP Security \(IPsec\)](#).
8. The Upper Layer Protocol header, which indicates which protocol a higher layer (such as the transport layer) is to process the packet with (for example, TCP).

## The Internet Control Message Protocol (ICMPv6)

The Internet Control Message Protocol, ICMPv6, provides a mechanism for error reporting and route discovery and diagnostics. It also conveys information about multicast group membership, a function that is carried out by the Internet Group Management Protocol (IGMP) in IPv4, and performs address resolution, which the Address Resolution Protocol (ARP) performs in IPv4.

Significant aspects of ICMPv6 include neighbour discovery, which allows one device in a network to find out about other nearby devices, and stateless address autoconfiguration, which allows a device to dynamically determine its own IPv6 address.

ICMPv6 is also used to support the Ping v6 (*Packet Internet Groper*) and Trace route v6 functions, which are used to verify the connections between networks and network devices. Ping is used to test the connectivity between two network devices to determine whether each network device can “see” the other device. Trace route is used to discover the route used to pass packets between two systems running the IP protocol.

Both of these functions operate almost identically in IPv4 and IPv6. For more information, see [“Ping and Trace Route” on page 23-38 of Chapter 23, Internet Protocol \(IP\)](#).

### Neighbour discovery

Neighbour discovery is an ICMPv6 function that allows a router or host to identify other devices on its links. This information is then used in address autoconfiguration, to redirect a node to use a more appropriate router if necessary, and to maintain reachability information with its neighbours.



The IPv6 Neighbour Discovery protocol is similar to a combination of the IPv4 protocols ARP, ICMP Router Discovery and ICMP Redirect.

The following table describes packet types involved with neighbour discovery.

Packet Type	Description
router solicitation	Packet in which a host sends out a request for routers to generate advertisements.
router advertisement	Allows routers to advertise their presence and other network parameters. A router sends an advertisement packet in response to a solicitation packet from a host.
neighbour solicitation	Packet in which a node sends a packet to determine the link layer address of a neighbour or to verify that a neighbour is still active.
neighbour advertisement	A response to a neighbour solicitation packet. These packets are also used to notify neighbours of link layer address changes.
redirect	Informs hosts of a better first hop.

To comply with Section 6.2.1 of RFC 2461, *IPv6 Neighbour Discovery*, the router does not generate router advertisements by default. See [“Neighbour Discovery” on page 34-16](#) for instructions about enabling advertisements.

The following table explains packet types and services.

Packet Type	Description
address resolution	A method for carrying out address autoconfiguration, and is achieved using the Neighbour Solicitation Message and the Neighbour Advertisement Message.
router and prefix discovery	<p>On connection to a link, a node needs to know the address of a router that the node can use to reach the rest of the world. The node also needs to know the prefix (or prefixes) that define the range of IP addresses on its link that it can reach without going through a router.</p> <p>Routers use ICMP to convey this information to hosts, by means of router advertisements. The message may have an option attached (the <i>source link address</i> option), which enables the receiving node to respond directly to the router, without performing a neighbour solicitation.</p>
immediate information	<p>The configuration of a router includes a defined frequency at which unsolicited advertisements are sent. If a node wants to obtain information about the nearest router immediately, rather than waiting for the next unsolicited advertisement, the node can send a router solicitation message.</p> <p>Each router that receives the solicitation message sends a router advertisement specifically to the node that sent the solicitation.</p>
redirection	If a node is aware of more than one router that it can use to connect to wider networks, the router to which it sends packets by default does not always represent the most desirable route. ICMPv6 uses the redirect packet to communicate a more effective path to the node.
Neighbour Unreachability Detection (NUD)	A node may issue solicitation requests to determine whether a path is still viable, or may listen in on acknowledgement packets of higher-layer protocols, such as TCP. If the node determines that a path is no longer viable, it attempts to establish a new link to the neighbour, or to re-establish the previous link. NUD can be used between any two devices in the network, independent of whether the devices are acting as hosts or routers.

### Stateless address autoconfiguration

Stateless address autoconfiguration allows an IPv6-aware device to be plugged into a network without manual configuration with an IP address. This plug and play functionality results in networks that are easier to set up and modify, and simplifies the process of shifting to use a new Internet Service Provider (ISP).

Stateless address autoconfiguration is achieved in a series of steps. Routers and hosts perform the first three steps, which autoconfigure a link-local address. A global address is autoconfigured in the last three steps, which only hosts perform.

#### On the router or host

1. During system start-up, the node begins autoconfiguration by generating a link-local address for the interface. A link-local address is formed by adding the interface ID to the link-local prefix fe80::/10.

2. The node then transmits a neighbour solicitation message to this address. If the address is already in use, the node that the address belongs to replies with a neighbour advertisement message. The autoconfiguration process stops and manual configuration of the node is then required.
3. If no neighbour advertisement is received, the node concludes that the address is available and assigns it to the chosen interface.

#### On the host

4. The node then sends one or more router solicitations to detect if any routers are present. Any routers present responds with a router advertisement.

If no router advertisement is received, the node tries to use DHCP to obtain an address and other configuration information. If no DHCP server responds, the node continues using the link-level address.

If a router advertisement is received, this message informs the node how to proceed with the auto configuration process.

5. The prefix from the router advertisement, if received, is added to the link-level address to form the global unicast IP address.
6. This address is then assigned to the network interface.

If routers are present, the node continues to receive router advertisements. The node updates its configuration when there are changes in the router advertisements.

## IPv6 Routing

Routing in IPv6 is almost identical to IPv4 routing under CIDR, except that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. For more information about routing, see [“Routing” on page 23-22 of Chapter 23, Internet Protocol \(IP\)](#).

### Routing Information Protocol (RIPv6)

RIP is a simple distance vector protocol that defines networks based on how many hops they are from the router. When a network is more than 15 hops away (one hop is one link), it is not included in the routing table.

RIPv6, also referred to as RIPv6 (for “next generation”) is similar to RIPv2, which is described in [“Routing Information Protocol \(RIP\)” on page 23-25 of Chapter 23, Internet Protocol \(IP\)](#). Extensions to RIPv2 to support IPv6 are:

- the address field of a routing entry is expanded to 128 bits to allow IPv6 prefixes
- the 32-bit RIPv2 subnet mask field is replaced by an 8-bit prefix length field
- authentication is removed in RIPv6
- the size of a routing packet is no longer arbitrarily limited
- RIPv6 specifies the next hop instead of simply allowing the recipient of the update to set the next hop to the sender of the update.

In RIPv6, each router uses a routing table to keep track of every destination that is reachable throughout the system. Each entry in the routing table contains:

- the IPv6 prefix of the destination
- a metric, which represents the total cost of getting a packet from the router to that destination

- the IPv6 address of the next router along the path to the destination
- a flag to indicate that information about the route has changed recently
- various timers associated with the route.

## IPv6 Filtering

With the increase in connections to the Internet, and the interconnection of networks from different organisations, filtering data packets is an important way to ensure that only legitimate connections are allowed. Security can never be perfect while connections to other networks exist, but filters let network managers manage permissible access while restricting users without permission.

IPv6 filtering uses a similar mechanism to IPv4 filtering. More information can be found in [“Policy-Based Routing” on page 23-26 of Chapter 23, Internet Protocol \(IP\)](#).

## Integration of IPv4 and IPv6

IPv6 has been designed in such a way that a smooth transition from IPv4 is possible. The most effective way to ensure this is to use a *dual IP stack*. A node configured as a dual stack system has both a 128-bit IPv6 address and a 32-bit IPv4 address, and can communicate with nodes running only IPv4 and nodes running only IPv6.

Another aspect of the transition period is the tunnelling of IPv6 packets across the IPv4 network. IPv6 packets are tunnelled simply by encapsulating the IPv6 packet within an IPv4 datagram, and identifying that this datagram is an encapsulated IPv6 packet by giving the datagram a protocol value of 41.

## IPv6 on the Switch

This section describes the switch’s support for IPv6, and how to configure IPv6 on the switch. Fundamental IPv6 features on the switch are:

- IPv6 interfaces and addresses
- extension header processing
- routing table processing
- RIPv6 (RIPng)
- neighbour discovery
- Stateless Address Autoconfiguration
- IPv6 filtering
- IPv6 fragmentation
- IPv6 multicasting, which is described in [Chapter 36, IPv6 Multicasting](#)
- IPsec and ISAKMP, which are described in [Chapter 49, IP Security \(IPsec\)](#)
- Dynamic Host Configuration Protocol (DHCP6), which is described in [Chapter 35, Dynamic Host Configuration Protocol for IPv6 \(DHCP6\)](#)

The switch also supports the following upper layer protocols:

- UDP, which transports RIPv6 packets
- TCP, which transports Telnet requests
- ICMPv6, which is used for stateless address autoconfiguration, neighbour discovery, Ping and Trace Route requests.

Integration of IPv6 with IPv4 is provided by:

- 6-to-4 support
- IPv6 static tunnelling.

Many of these features are performed automatically by the switch, and most commands operate in a similar manner to their IPv4 equivalents.

## Enabling IPv6

The switch's implementation of IPv6 is disabled by default. To enable IPv6, use the command:

```
enable ipv6
```

To disable IPv6, use the command:

```
disable ipv6
```

Any IPv6 configuration that the switch has performed dynamically is preserved between disabling and re-enabling IPv6. For example, addresses that have been configured are still present.

To display information about IPv6 settings, use the command:

```
show ipv6
```

To display IPv6 counters, use the command:

```
show ipv6 counter
```

Because the switch implements IPv6 as a dual stack, implementing IPv6 does not affect IPv4 functionality.

## IPv6 Interfaces and Addresses

The switch supports the addition of IPv6 addresses directly to Point-to-Point Protocol (PPP) and Virtual Local Area Network (VLAN) interfaces and indirectly to virtual interfaces, when the tunnel is created.

As with IPv4 addresses, a proportion of the leftmost bits of the IPv6 address can be the address of a subnet, rather than a host. This subnet part of the address is called the *prefix*. The prefix is specified by following the IPv6 address with a slash (/) and the length of the prefix in bits:

```
ipv6address=ipv6add/prefix-length
```

This syntax is referred to as *slash notation*.

To create an IPv6 logical interface and associate it with an interface, use the command:

```
create ipv6 interface=interface [duptrans=1..16]  
[retrans=0..4294967295]
```

where *interface* is an interface name formed by concatenating an interface and an interface instance (such as ppp1, vlan1).

As part of the creation process, the switch performs stateless address autoconfiguration to assign an IPv6 address to the interface, by adding the interface's MAC address to the reserved IPv6 prefix fe80::.. These addresses are link-local addresses, which are sufficient for communication among devices on the same link. Several interfaces can be given the same link-local address, as specified in RFC 2373, *IP Version 6 Addressing Architecture*.

The **duptrans** parameter sets the number of neighbour solicitation messages that the switch sends during the Duplicate Address Detection process. The **retrans** parameter sets the number of times the switch resends each Router Advertisement message. Lossy links can be partially compensated for by increasing these counters.

If the switch is used as a host instead of a router, it is still able to use stateless address autoconfiguration to generate a link-local address. However, it is unable to receive router advertisements, and therefore unable to autoconfigure a global address. If it is necessary to give the switch a global IPv6 address, it must be added manually.

To add another IPv6 address manually to the interface, or to create the interface and manually add the IPv6 address to it at the same time, use the command:

```
add ipv6 interface=interface
    ipaddress=ipv6add/prefix-length [filter=0..99]
    [preferred=1..4294967295|infinite]
    [priorityfilter=200..299] [publish={yes|no}]
    [type={anycast|unicast}] [valid=1..4294967295|infinite]
```

An interface can have only one link-local address at a time. If a new link-local address is added on an IPv6 interface, the interface uses the new link-local address. The link-local address of a static tunnel cannot be changed after the tunnel has been created (see [“Static tunnelling” on page 34-21](#) for information about tunnels). The link-local address of an interface cannot be changed while PIM6 is attached to it. The PIM6 interface must first be deleted by using the command:

```
delete pim6 interface in Chapter 36, IPv6 Multicasting
```

If the address is an anycast address (see [“Anycast addresses” on page 34-5](#)), the **type** parameter allows it to be distinguished from a unicast address.

To change the address or other parameters of the interface, use the command:

```
set ipv6 interface=interface
    ipaddress=ipv6add/prefix-length [filter=0..99]
    [preferred=1..4294967295|infinite]
    [priorityfilter=200..299] [publish={yes|no}]
    [valid=1..4294967295|infinite]
```

To destroy an IPv6 interface, use the command:

```
destroy ipv6 interface=interface
```

To delete an address from an interface, use the command:

```
delete ipv6 interface=interface ipaddress=ipv6add
```

Deleting a link-local address on an interface causes the interface to revert back to its original auto-configured link-local address. An interface's auto-

configured IPv6 link-local address cannot be deleted except by destroying the interface.

To display information about the configured interfaces, use the command:

```
show ipv6 interface [=interface]
```

To display information about IPv6 multicast addresses, use the command:

```
show ipv6 multicast
```

To associate an IPv6 address with the name of a host, use the command:

```
add ipv6 host=name ipaddress=ipv6add
```

This functionality is similar to the [add ip host command on page 23-73 of Chapter 23, Internet Protocol \(IP\)](#).

To disassociate the IPv6 address and the host name, use the command:

```
delete ipv6 host=name
```

To display information about the host name table, use the command:

```
show ipv6 host
```

## Extension Header Processing

All routers in the path to the final destination process the routing header and the hop by hop header to route the packet to the specified path. The final node processes the fragment header. When the switch is the source or final destination of the packet, it processes these extension headers as required.

## Routing Table Processing and RIPv6

The switch maintains and processes a routing table for IPv6 addresses, in a similar manner to the IPv4 routing table described in [“Routing” on page 23-22 of Chapter 23, Internet Protocol \(IP\)](#).

The Router Information Protocol (RIPv6 or RIPng, as described in RFC 2080, *RIPng for IPv6*) is supported, and allows the switch to share information from its routing tables with routers that connect other networks. RIPv6 passes routing table information from neighbour to neighbour along a line of routers.

RIPv6 packets are transported over UDP. When IPv6 is enabled, the switch can route UDP packets through an IPv6 network or tunnel.

RIPv6 routing is disabled by default. To enable it, use the command:

```
enable ipv6 rip
```

To enable receiving or sending of RIP packets on an interface, use the command:

```
add ipv6 rip interface=interface poisonreverse={on|off}
```

This command also specifies whether poison reverse is enabled on the interface. Poison reverse addresses the problem of slow convergence on RIPng routes. If one device in a network goes down, it can take a long time for the devices to recognise that routes through the crashed device are no longer available. With poison reverse, when a device goes down, the devices next to it

continues to advertise the route but with a cost of 16. This cost indicates that the route is unavailable.

To stop an interface from sending or receiving RIP packets, use the command:

```
delete ipv6 rip interface=interface
```

To disable RIP, use the command:

```
disable ipv6 rip
```

To display information about RIP counters or timers, use the command:

```
show ipv6 rip [counter|timer]
```

Under some conditions, the route that is dynamically determined by RIPv6 may not be the most desirable one. For example, certain packets may need to be sent over a more secure route. To statically add the desired route to an interface, use the command:

```
add ipv6 route=ipv6add/prefix-length interface=interface  
nexthop=ipv6add [metric=1..16] [preference=0..65535]
```

The **route** parameter identifies the final destination network to which the packets are sent. This command can also add a default route where packets can be sent when no other route is found, for example, the gateway between the LAN and the wider network. To add a default route, use the command:

```
add ipv6 route=::/0 interface=interface  
nexthop=ipv6add-to-send-packets-to  
[metric=1..16] [preference=0..65535]
```

To delete a route from an interface, use the command:

```
delete ipv6 route=ipv6add interface=interface nexthop=ipv6add
```

To display information about the IPv6 routes on the switch, use the command:

```
show ipv6 route [full]
```

## Neighbour Discovery

The switch issues router advertisement messages in response to a router solicitation message from a host so the host can determine the switch's identity and availability. The switch also sends neighbour solicitation messages to neighbouring nodes and responds to neighbour solicitation messages with a neighbour advertisement message. The address resolution mechanism and queue structure is similar to that used in ARP in IPv4. More information about neighbour discovery can be found in RFC 2461, *Neighbour Discovery for IPv6*.

To comply with Section 6.2.1 of RFC 2461, *IPv6 Neighbour Discovery*, the switch does not generate router advertisements by default. To enable the switch to generate router advertisements for all interfaces, use the command:

```
enable ipv6 advertise
```

To enable the switch to generate router advertisements for a particular interface, use the command:

```
enable ipv6 advertise interface=interface
```



For each IPv6 interface you want to have advertised, also specify that the interface's prefix should be included in advertisements, using one of the commands:

```
add ipv6 interface=interface ipaddress={ipv6add/prefix
length|dhcp|dhcptemp|pd} publish=yes

set ipv6 interface=interface publish=yes
```

If the switch does not generate router advertisements, the router discovery and address autoconfiguration processes in the network may not work as expected.

To disable router advertisements, use the command:

```
disable ipv6 advertise [interface=interface]
```

Neighbour discovery and solicitation are part of the IPv6 protocol and cannot be disabled.

To set neighbour discovery and Router Advertisement counters and timers, use the command:

```
set ipv6 nd interface=interface [duptrans=1..16] [hop=1..255]
[life=0|4..9000] [maxaint=4..1800] [mconf={yes|no}]
[minaint=3..1350] [mtu=1280..65535] [oconf={yes|no}]
[reach=0..3600000] [retrans=0..4294967295]
```

To add a prefix to Router Advertisements for a particular interface, without adding the prefix to the interface's IPv6 address, use the command:

```
add ipv6 prefix=ipv6add/prefix-length interface=interface
[autonomous={yes|no}] [onlink={yes|no}]
[preferred=1..4294967295|infinite] [valid=1..4294967295|
infinite]
```

To modify a prefix in the Router Advertisement prefix list, use the command:

```
set ipv6 prefix=ipv6add/prefix-length interface=interface
[autonomous={yes|no}] [onlink={yes|no}]
[preferred=1..4294967295|infinite] [valid=1..4294967295|
infinite]
```

To display information about the neighbours determined by neighbour discovery, use the command:

```
show ipv6 ndcache
```

To display information about the neighbour discovery parameters, and the list of prefixes that are included in Router Advertisements, use the command:

```
show ipv6 ndconfig
```

## IPv6 Filtering

IPv6 packets can be filtered on arrival at the switch, with a traffic filter, or on transmission from the switch, with a priority filter. Traffic filters can be used to determine whether an incoming packet is accepted or rejected. Priority filters apply a particular priority to the packets. Priorities range from 0 to 7, with 0 having the highest priority.

To add an IPv6 filter, use the command:

```
add ipv6 filter=filter-id
source=ipv6add/prefix-length [action={include|exclude}|
priority=p0..p7] [destination=ipv6add/prefix-length]
[dport={port-name|port-id|any}] [entry=entry-number]
[icmpcode={icmp-code-name|icmp-code-id|any}]
[icmptype={icmp-type-name|icmp-type-id|any}]
[log={4..1950|dump|header|none}] [options={yes|no|on|off|
enable|disable}] [protocol={protocol|any|egg|icmp|ospf|
tcp|udp}] [session={any|established|start}] [size={size|
any}] [sport={port-name|port-id|any}]
```

The **add ipv6 filter** command can be used to add another entry to an existing filter, by identifying the filter by its filter-id and using the **entry** parameter to indicate the desired entry number for the new entry. Entries with the lowest entry numbers are higher in the filter and are applied to traffic before higher-numbered entries.

To modify an existing filter entry, use the command:

```
set ipv6 filter=filter-id
entry=entry-number [source=ipv6add/prefix-length]
[action={include|exclude}|priority=p0..p7]
[destination=ipv6add/prefix-length] [dport={port-name|
port-id|any}] [icmpcode={icmp-code-name|icmp-code-id|
any}] [icmptype={icmp-type-name|icmp-type-id|any}]
[log={4..1950|dump|header|none}] [options={yes|no|on|off|
enable|disable}] [protocol={protocol|any|egg|icmp|ospf|
tcp|udp}] [session={any|established|start}] [size={size|
any}] [sport={port-name|port-id|any}]
```

To delete a filter or one of its entries, use the command:

```
delete ipv6 filter=filter-id ENTRY={entry-number|ALL}
```

To display information about existing filters, sorted by filter and entry numbers, use the command:

```
show ipv6 filter[=filter-id]
```

## IPv6 Fragmentation

Routers along a path send an “ICMP packet too big” reply when they receive a packet that is larger than the MTU for the link. It is then up to the host to fragment the packets.

When the switch is acting as a host, it performs fragmentation as needed. If the packet size is greater than the MTU the packet is fragmented into packets that are sized in multiples of 8 bytes. Each of the new packets carries a fragmentation header. A packet with a fragmentation header is identified with the value 44 in the previous header’s Next Header field.

## Telnet v6

The **telnet** command allows remote access to a device’s command-line interface. It operates similarly in IPv4 and IPv6. For more information, see “Telnet” on page 62-6 of [Chapter 62, Terminal Server](#).

To Telnet to an IPv6 interface with the address 1111::260:0:97ff:64aa, use the command:

```
telnet 1111::260:0:97ff:64aa
```

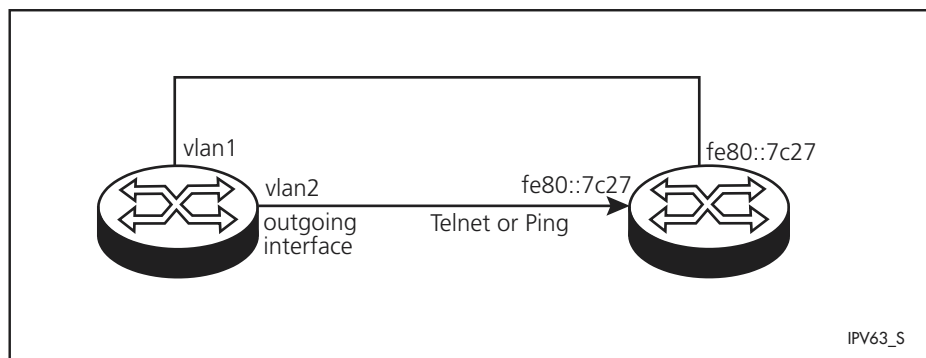
Telneting to a link-local address requires interface information as well as the address, because a single link-local address can belong to several interfaces. To Telnet to a link-local address, specify the interface out which the switch is to send the Telnet request, as well as the address to which the switch is to send the Telnet request (Figure 34-2). The Telnet syntax is:

```
telnet ipv6-address%interface
```

For example:

```
telnet fe80::7c27%vlan1
```

Figure 34-2: The outgoing interface to specify when Pinging, Telneting or sending an SSH request to an IPv6 link-local address



Instead of entering the IPv6 address of the host's interface, an easy-to-remember name can be associated with the host by using the command:

```
add ipv6 host=name ipaddress=ipv6add
```

Telnet messages are transported over TCP. When IPv6 is enabled, the switch can route TCP packets through an IPv6 network or tunnel.

## Ping

The switch supports an extended [ping command on page 23-121 of Chapter 23, Internet Protocol \(IP\)](#), which allows the user to attempt to contact an IPv6 interface and record whether packets are received and the response time if they are. Defaults, including a default address, can be entered with the [set ping command on page 23-154 of Chapter 23, Internet Protocol \(IP\)](#).

To ping an IPv6 interface with the address 1111::260:0:97ff:64aa, use the command:

```
ping 1111::260:0:97ff:64aa
```

Pinging a link-local address requires additional information, because a single link-local address can belong to several interfaces. To ping a link-local address, specify the interface out which the switch is to send the ping request, as well as the address to which the switch is to send the ping request (Figure 34-2 on [page 34-19](#)). The ping syntax is:

```
ping ipv6-address%interface
```

For example:

```
ping fe80::7c27%vlan1
```

If a source IPv6 address is also specified with the **sipaddress** parameter, the source address must be on the outgoing interface. The **sipaddress** cannot be a link-local address.

## Secure Shell

The **ssh** command on page 46-38 of Chapter 46, *Secure Shell* initiates a Secure Shell connection to the specified host.

A Secure Shell request to an IPv6 link-local address requires interface information as well as the address, because a single link-local address can belong to several interfaces. To open a secure shell session to a link-local address, specify the interface out which the switch is to send the secure shell request, as well as the address to which the switch is to send the request (Figure 34-2 on page 34-19). For example:

```
ssh fe80:7c27%vlan1 user=Admin password=18Again
```

## Trace Route

The **trace** command on page 23-213 of Chapter 23, *Internet Protocol (IP)* records the path to an IPv6 node.

To view the route to an IPv6 node, use the command:

```
trace ipv6add
```

## Tunnelling IPv6 Packets over IPv4

This section describes options for tunnelling IPv6 packets through an IPv4 network:

- **6-to-4**
- **Static tunnelling**

The maximum number of simultaneous IPv6 tunnels available on the switch is 100. Static IPv6 tunnels and 6-to-4 tunnels share this resource. For example, a switch operating 60 static tunnels will have 40 free tunnels for 6-to-4 tunnelling.

- 6-to-4** The switch can automatically tunnel IPv6 packets over IPv4 interfaces, as described in RFC 3056 - *Connection of IPv6 Domains via IPv4 Clouds*. The interfaces must first be correctly configured with global IPv4 addresses, and routes must be set up to ensure the switch can communicate with the router at the other end of the tunnel using IPv4.

To configure a 6-to-4 interface, use the command:

```
add ipv6 6to4 ip=ipv4add
```

where *ipv4add* is the global IPv4 address of the IPv4 interface that IPv6 packets are transmitted and received on.

This command creates a virtual tunnel interface, with an interface name of *virt*, and assigns it a link-local IPv6 address. The first tunnel created is numbered *virt0*, and each succeeding tunnel is given the next available instance number. Each virtual interface behaves as a normal IPv6 interface. The interface is given a link-local IPv6 address with the prefix 2002::/16, of the form:

```
2002:<ipv4-address>::<ipv4-address>
```

For example, if a PPP interface with the address 203.109.0.1 is configured as a 6-to-4 address, the IPv6 address of the *virt0* interface is:

```
2002:cb6d:0001::cb6d:0001
```

To forward the desired IPv6 packets through the tunnel, add a route by using the command:

```
add ipv6 route=host-ipv6add/prefix-length
int=6to4-tunnel-interface nexthop=ipv6add
```

where *host-ipv6add* is the IPv6 address of the destination host or router, *ipv6add* is the link-local address of the 6-to-4 interface on the switch at the other end of the tunnel, and *6to4-tunnel-interface* is the VIRT interface that the switch created (such as *virt0*).

To display tunnel interface names, instances, IPv6 addresses, and other information for all tunnels that are configured on the switch, use the command:

```
show ipv6 tunnel
```

To display information about a specific tunnel, use the command:

```
show ipv6 interface=tunnel-interface
```

To remove a 6-to-4 interface, use the command:

```
delete ipv6 6to4 IP=ipv4add
```

Duplicate Address Detection (DAD) is not performed on IPv6 tunnels.

For a configuration example, see [“Dynamic \(6-to-4\) Tunnelling over an IPv4 Network” on page 34-29](#).

## Static tunnelling

As well as 6-to-4 automatic tunnelling, IPv6 subnets can be linked over an IPv4 tunnel by configuring static tunnels. To link two IPv6 networks through an IPv4 tunnel, first create the tunnel on each switch by using the command:

```
add ipv6 tunnel local=ipv4add target=ipv4add
[interface=tunnel-interface] [ipaddress=ipv6add]
```

This command creates a virtual tunnel interface, with an interface name of *virt*, and assigns it a link-local IPv6 address. To specify an interface instance for the tunnel, enter it using the **interface** parameter (for example, *virt5*). Otherwise, the first tunnel created is numbered *virt0*, and each succeeding tunnel is given the next available instance number. Each virtual interface behaves as a normal IPv6 interface. The **local** parameter is the IPv4 address of the interface through which packets are sent and received on the local switch. The **target** parameter is the IPv4 address of the remote switch's interface. These IPv4 interfaces can be PPP or VLAN interfaces.

The tunnel can be given an IPv6 address manually, by specifying the optional parameter, **ipaddress**. If this parameter is not specified, the link-local address assigned to the tunnel interface is:

```
fe80::<local-ipv4-address>
```

where *local-ipv4-address* is in hexadecimal. For example, if one end of a tunnel is created by using the command:

```
add ipv6 tunnel local=192.168.1.2 target=192.168.1.1
```

then the interface's link-local address is fe80::c0a8:0102.

Once the tunnel has been created, add a route on each interface to direct traffic to the other IPv6 network through the tunnel by using the command:

```
add ipv6 route=ipv6add interface=tunnel-interface
```

where *ipv6add* is the address of the IPv6 network on the other switch (not the IPv6 address of the tunnel), and *tunnel-interface* is the interface that the tunnel was given when it was created (e.g. virt0).

To display tunnel interface names, instances, IPv6 addresses, and other information for all tunnels that are configured on the switch, use the command:

```
show ipv6 tunnel
```

To display information about a specific tunnel, use the command:

```
show ipv6 interface=tunnel-interface
```

To delete a tunnel from an interface, use the command:

```
delete ipv6 tunnel=ipv6add interface=tunnel-interface
```

The IPv6 address in the **tunnel** parameter of the **delete** command is the original link-local address of the tunnel (the address that the switch assigned to the tunnel when it was created).

For a configuration example, see [“Static Tunnelling over an IPv4 Network” on page 34-31](#).

## Configuration Examples

This section has the following examples of how to configure IPv6 on the switch:

- [Basic Routing](#)
- [Dynamic Routing with RIPv6](#)
- [Dynamic \(6-to-4\) Tunnelling over an IPv4 Network](#)
- [Static Tunnelling over an IPv4 Network](#)
- [IPv6 Filters](#)

Before an IPv6 address can be added to a VLAN, the VLAN must be created, using the **create vlan** command. For more information about this command or VLANs, see [Chapter 8, Switching](#).

### Basic Routing

This example demonstrates configuring a switch's VLAN interface with an IPv6 address, and enabling it to access local hosts and external hosts through a gateway device. The gateway device and any other routers that the switch needs to communicate with must also be configured with appropriate IPv6 interfaces, addresses, and routes.

#### To configure basic IPv6 routing on the switch

##### 1. Enable the IPv6 module.

Enable IPv6 by using the command:

```
enable ipv6
```

##### 2. Add a global IPv6 address to an interface.

To manually add the IPv6 address 3FFE::1/32 to the interface vlan1, use the command:

```
add ipv6 interface=vlan1 ipaddress=3FFE::1/32
```

##### 3. Check the automatically-created route.

When an interface is added, an interface route is automatically created. Check this route by displaying the routing information by using the command:

```
show ipv6 route full
```

Figure 34-3: Example output from the **show ipv6 route** command for a basic IPv6 network

```

Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel  DLCI  Flags
-----
3ffe::/32 --->  ::
vlan1  no  0      interface 1      0    no    -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown

```

The switch can now communicate with other hosts on the network with the same prefix. To test that this route is functional, use the **ping** command to access another node in the local network:

```
ping 3ffe::2
```

Replies from the node are echoed on screen.

**4. Add a default route to the local gateway.**

The local gateway is the device connecting the local network to other networks such as the Internet. The default route is the route to which a packet is sent when the switch cannot find a route for it. To add a default route, set the **route** parameter to `::/0` and the **nexthop** parameter to the address where the packets should be sent.

To add a default route, when the switch is connected to the local LAN via the `vlan1` interface and the local gateway's IPv6 address is `3FFE::2`, use the command:

```
add ipv6 route=::/0 next=3ffe::2 interface=vlan1
```

**5. Test this route with ping.**

Once a return route has been created from the network `5FFE::/32` to `3FFE::/32`, test communications between the networks by using the command:

```
ping 5ffe::1
```

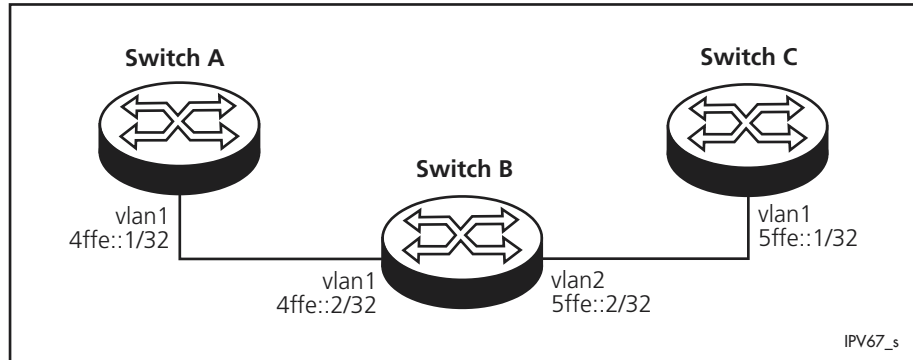
The switch can now access all local hosts with the same prefix, and to any host that the gateway device has a route.



## Dynamic Routing with RIPv6

RIP works between switches connected to multiple networks. This example demonstrates configuring three switches' VLAN interfaces with IPv6 addresses, and enabling RIPv6 routing on them so the switches at each extreme end can route to networks to which the others have access. This configuration is shown in the following figure.

Figure 34-4: Example for dynamic routing



### To configure dynamic routing with RIPv6

#### 1. Enable the IPv6 module.

Enable IPv6 by using the following command on each switch:

```
enable ipv6
```

#### 2. Create the IPv6 interfaces and add global IPv6 addresses to each required interface.

Manually add IPv6 addresses to Ethernet interfaces on each switch. On switch A, use the command:

```
add ipv6 interface=vlan1 ipaddress=4ffe::1/32
```

On switch B, use the commands:

```
add ipv6 interface=vlan1 ipaddress=4ffe::2/32
```

```
add ipv6 interface=vlan2 ipaddress=5ffe::2/32
```

On switch C, use the command:

```
add ipv6 interface=vlan1 ipaddress=5ffe::1/32
```

The switch automatically assigns a link local address to each interface.

#### 3. Check the routes on each switch.

When the interfaces are added, interface routes are automatically created. Check these routes by displaying the routing information, using the following command on each switch:

```
show ipv6 route full
```

This command produces output similar to the following figures.

Figure 34-5: Output on switch A showing the interface route

```
IPV6 Routing Table Entries
Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel DLCI Flags
-----
4ffe::/32 ---> ::
vlan1 no 0          interface 1      0    no    -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown
```

Figure 34-6: Output on switch B showing the interface routes

```
IPV6 Routing Table Entries
Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel DLCI Flags
-----
4ffe::/32 ---> ::
vlan1 no 0          interface 1      0    no    -
5ffe::/32 ---> ::
vlan2 no 0          interface 1      0    no    -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown
```

Figure 34-7: Output on switch C showing the interface route

```
IPV6 Routing Table Entries
Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel DLCI Flags
-----
5ffe::/32 ---> ::
vlan1 no 0          interface 1      0    no    -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown
```

**4. Enable RIP on each switch and add a RIP interface.**

To enable RIP, use the following command on each switch:

```
enable ipv6 rip
```

To create an RIP interface on the network between the switches, use the following commands:

On switch A:

```
add ipv6 rip interface=vlan1
```

On switch B:

```
add ipv6 rip interface=vlan1
```

```
add ipv6 rip interface=vlan2
```

On switch C:

```
add ipv6 rip interface=vlan1
```

The RIP updates take 30 seconds to propagate between the switches. After this time, switch A should contain a route to switch C, and switch C a route to switch A.

**5. Check the routes.**

On each switch, display the routes to check that the routes appear on the opposite switch by using the command:

```
show ipv6 route full
```

Output on switch A should be similar to the following figure.

Figure 34-8: Output on switch A

```
IPV6 Routing Table Entries
Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel DLCI Flags
-----
5ffe::/32 ---> fe80::0200:cdff:fe00:a14d
vlan1 yes 0      ripng      2      100 no      -
4ffe::/32 ---> ::
vlan1 no  0      interface 1      0      no      -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown
```

Figure 34-9: Output on switch C

```
IPV6 Routing Table Entries
Destination prefix ---> Next Hop
Int.  Age Policy Protocol  Metric Pref Tunnel DLCI Flags
-----
4ffe::/32 ---> fe80::0200:cdff:fe00:a14d
vlan1 yes 0      ripng      2      100 no      -
5ffe::/32 ---> ::
vlan1 no  0      interface 1      0      no      -
-----
Codes: P=publish, D=default, A=addrconf, S=stale, L=onlink
N=nonexthop, C=cache, F=flow, U=unknown
```

Note that the next hop for the RIPv6 routes is the link-local address for the other switch, not the 4FFE::/32 or 5FFE::/32 address. This allows switches to share routes even if they aren't on the same logical network. In

this example, `fe80::0200:cdf:fe00:a14d` is the link-local address of the vlans on switch B.

**6. Test these routes with ping.**

On switch A, use the command:

```
ping 5ffe::1
```

On switch C, use the command:

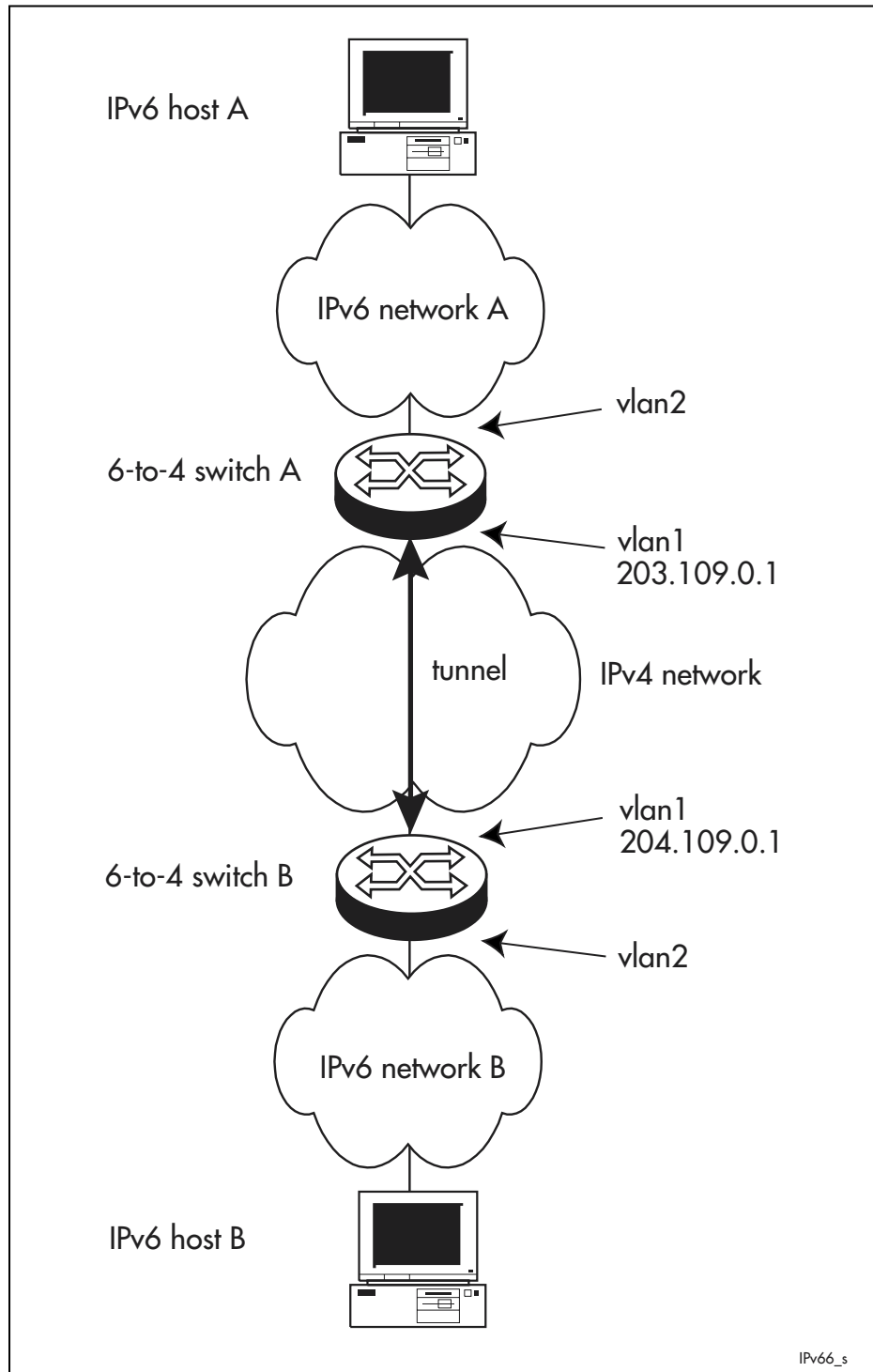
```
ping 4ffe::1
```

Each switch can now communicate with the networks to which they previously did not have a route.

## Dynamic (6-to-4) Tunnelling over an IPv4 Network

A 6-to-4 tunnel allows IPv6 packets to be routed between IPv6-aware nodes that are connected by an IPv4 network. The switch automatically encapsulates IPv6 packets. This configuration is shown in the following figure.

Figure 34-10: Example of a dynamic tunnel



## To configure a 6-to-4 tunnel

### 1. Enable IP and add an interface to each switch.

On switch A, use the commands:

```
enable ip
add ip interface=vlan1 ipaddress=203.109.0.1
```

On switch B, use the commands:

```
enable ip
add ip interface=vlan1 ipaddress=204.109.0.1
```

### 2. Set up IPv4 routes to connect the switches.

On switch A, use the command:

```
add ip route=0.0.0.0 interface=vlan1 nexthop=204.109.0.1
```

On switch B, use the command:

```
add IP route=0.0.0.0 interface=vlan1 nexthop=203.109.0.1
```

### 3. Enable IPv6 on both switches.

Once the IPv4 connection is working correctly, enable IPv6 on each switch by using the command:

```
enable ipv6
```

### 4. Configure the 6-to-4 interfaces.

On switch A, use the command:

```
add ipv6 6to4 ip=203.109.0.1
```

The interface's name is virt0. It has a link-local IPv6 address of 2002:cb6d:1::cb6d:1

On switch B, use the command:

```
add ipv6 6to4 ip=204.109.0.1
```

The interface's name is virt0. It has a link-local IPv6 address of 2002:cc6d:1::cc6d:1

### 5. Set up routes to forward packets via the tunnel.

On switch A, use the command:

```
add ipv6 route=::/0 interface=virt0
nexthop=2002:cc6d:1::cc6d:1
```

On switch B, use the command:

```
add ipv6 route=::/0 interface=virt0
nexthop=2002:cb6d:1::cb6d:1
```

### 6. Check the configuration.

Use the commands:

```
show ipv6 interface=virt0
show ipv6 tunnel
```

Figure 34-11: Output of the **show ipv6 interface=virt0** command for switch A

```

IPV6 Interface Configuration
-----
Interface ..... virt0
Ipv6 Interface Index ..... 3
Link-layer address ..... 6-to-4 interface
EUI-64 Interface Identifier ..... 6-to-4 interface
IPSec ..... No
True MTU/Link MTU ..... -/1280
Multicast status ..... Enabled
Send Router Advertizements ? ..... No
Ipv6 Interface Addresses :
  Int  Addresses
  Type  Scope  State      Enabled      PLen      Preferred
-----
  1     2002:cb6d:0001::cb6d:0001
      unicast  global preferred  Yes         /16        infinite

```

Figure 34-12: Output of the **show ipv6 tunnel** command for switch A

```

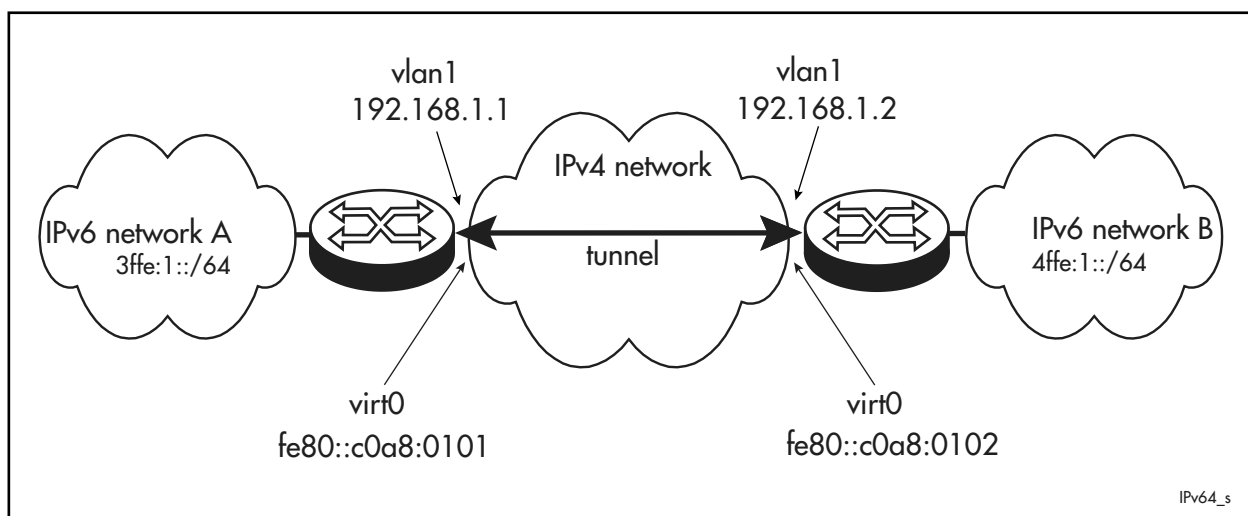
Ipv6 Tunnels:
Ipv6 Tunnel Address
  Tunnel start      Tunnel end
-----
  1  2002:cb6d:0001::cb6d:0001
    6-to-4 interface
-----

```

## Static Tunnelling over an IPv4 Network

A static tunnel allows IPv6 packets to be routed between IPv6-aware nodes that are connected by an IPv4 network. The configuration is shown in the following figure.

Figure 34-13: Example of a static tunnel



The following table explains the interfaces and addresses in this example.

Switch	IPv4 address	IPv6 prefix of IPv6 network
A	192.168.1.1	3ffe:1::/64
B	192.168.1.2	4ffe:1::/64

### To configure a static tunnel

#### 1. Enable IP and add an interface to each switch.

Details on configuring basic IPv4 routing can be found in [“Basic IP Setup over PPP” on page 23-49 of Chapter 23, Internet Protocol \(IP\)](#).

On switch A, use the commands:

```
enable ip
add ip interface=vlan1 ipaddress=192.168.1.1
```

On switch B, use the commands:

```
enable ip
add ip interface=vlan1 ipaddress=192.168.1.2
```

#### 2. Enable IPv6 on both switches.

Once the IPv4 connection is working correctly, enable IPv6 on each switch by using the command:

```
enable ipv6
```

#### 3. Create a tunnel between the two switches.

On switch A, use the command:

```
add ipv6 tunnel local=192.168.1.1 target=192.168.1.2
```

This command creates an IPv6 interface (virt0 if this is the first tunnel created), with link-local address fe80::c0a8:0101.

On switch B, use the command:

```
add IPV6 tunnel local=192.168.1.2 target=192.168.1.1
```

This command creates an IPv6 interface (virt0 if this is the first tunnel created), with link-local address fe80::c0a8:0102.

The **local** parameter is the IPv4 address of the switch at the local end of the tunnel, so for switch A the **local** parameter is A's IPv4 address and the **target** parameter is B's IPv4 address. For switch B the **local** parameter is B's IPv4 address and the **target** parameter is A's IPv4 address.

#### 4. Create a route for the tunnel.

Add a route on each switch, pointing to the IPv6 network on the other switch.

On switch A, use the command:

```
add ipv6 route=4ffe:1::/64 int=virt0
```

On switch B, use the command:

```
add ipv6 route=3ffe:1::/64 int=virt0
```

Note that the **nexthop** parameter is not necessary for a tunnel.

#### 5. Test the routes.

On switch A, use the command:

```
ping fe80::c0a8:0102%virt0
```



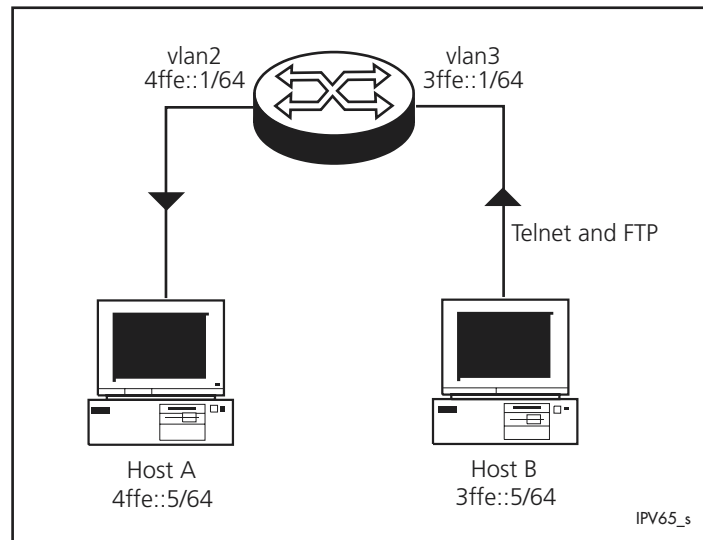
On switch B, use the command:

```
ping fe80::c0a8:0101%virt0
```

## IPv6 Filters

This example illustrates how to configure IPv6 filters on the switch. There are two hosts and Host A needs to be accessed by Telnet and FTP from Host B. Other traffic from B to A is blocked by the filter, except neighbour discovery, which is required for basic IPv6 routing. This configuration is shown in the following figure.

Figure 34-14: Example showing host and switch connections



### To configure the required IPv6 filters

#### 1. Enable IPv6.

Enable IPv6 by using the command:

```
enable ipv6
```

#### 2. Create the interfaces and assign IPv6 addresses to them, using the commands:

```
create vlan=marketing vid=2
add vlan=2 port=2
create vlan=admin vid=3
add vlan=3 port=3
add ipv6 int=vlan2 ip=4ffe::0001/64
add ipv6 int=vlan3 ip=3ffe::0001/64
```

**3. Create the filter and add entries to it to allow Telnet and FTP traffic.**

To create the filter and add filter entries to enable Telnet and FTP access to Host A from Host B, use the commands:

```
add ipv6 filter=1 entry=1 source=3ffe::0005
    destination=4ffe::0005 protocol=tcp sport=any dport=23
    log=header

add ipv6 filter=1 entry=2 source=3ffe:0005
    destination=4ffe:0005 protocol=tcp sport=any dport=21
    log=header session=any

add ipv6 filter=1 entry=3 source=4ffe::0005
    destination=3ffe::0005 session=any protocol=tcp
    sport=any dport=ftpdata log=header
```

**4. Add entries to the filter to allow Neighbour Discovery traffic.**

In addition to the Telnet and FTP filter entries, it is necessary to explicitly allow IPv6 Neighbour Discovery (ND) traffic from Host B to Host A, because this traffic is required to enable basic connectivity between devices. The following filter entries allow the types of ICMP traffic that are typically IPv6 ND traffic to pass through the switch:

```
add ipv6 filter=1 entry=4 source=3ffe::/64 protocol=icmp
    icmptype=135 icmpcode=any

add ipv6 filter=1 entry=5 source=3ffe::/64 protocol=icmp
    icmptype=136 icmpcode=any
```

**5. Assign the filter to the switch's interface to Host B.**

```
set ipv6 int=vlan3 ipv6=3ffe::0001/64 filter=1
```

**6. Check the configuration.**

The definitions of the filters on the switch can be checked with the command:

```
show ipv6 filter
```

The following command displays details of the IP interfaces defined:

```
show ipv6 interface
```

## Command Reference

---

This section describes the commands available to configure and manage IPv6 on the switch.

The shortest valid command is denoted by capital letters in the Syntax section. See [“Conventions” on page lxvi of About this Software Reference](#) for details of the conventions used to describe command syntax. See *Appendix A, Messages* for a complete list of messages and their meanings.

### add ipv6 6to4

---

**Syntax** `ADD IPV6 6TO4 IP=ipv4add`

where *ipv4add* is a valid IPv4 address in dotted decimal notation

**Description** This command configures the switch to recognise 6-to-4 addresses and allows the switch to act as a 6-to-4 relay. The switch then performs automatic 6-to-4 tunnelling with the correct route configuration.

The **ip** parameter specifies the IPv4 address of an IPv4 interface on the switch.

**Example** To create a 6-to-4 tunnel, for traffic over an interface with the IPv4 address 202.101.202.101, use the command:

```
add ipv6 6to4 ip=202.101.202.101
```

**Related Commands** [add ipv6 tunnel](#)  
[delete ipv6 6to4](#)  
[show ipv6 tunnel](#)

## add ipv6 filter

**Syntax** ADD IPV6 FILTER=*filter-id* SOURCE=*ipv6add/prefix-length*  
 [ACTION={INCLUDE|EXCLUDE}|PRIORITY=P0..P7]  
 [DESTINATION=*ipv6add/prefix-length*] [DPORT={*port-name*|*port-id*|ANY}] [ENTRY=*entry-number*]  
 [ICMPCODE={*icmp-code-name*|*icmp-code-id*|ANY}]  
 [ICMPTYPE={*icmp-type-name*|*icmp-type-id*|ANY}]  
 [LOG={4..1950|Dump|Header|None}] [OPTIONS={YES|NO|ON|OFF|ENABLE|DISABLE}] [PROTOCOL={*protocol*|Any|Egp|Icmp|Ospf|Tcp|Udp}] [SESSION={Any|Established|Start}] [SIZE={*size*|ANY}] [SPORT={*port-name*|*port-id*|ANY}]

where:

- *filter-id* is a number from 0 to 99 or 200 to 299.
- *ipv6add* is a valid IPv6 address, with its prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is an integer from 1 to 128.
- *port-name* is the predefined name for a TCP or UDP port.
- *port-id* is an IP port number or a range of port numbers in the format [*low*]:[*high*].
- *entry-number* is the position of this entry in the filter.
- *icmp-code-name* is the predefined name for an ICMP reason code.
- *icmp-code-id* is the number of an ICMP reason code.
- *icmp-type-name* is the predefined name for an ICMP reason type.
- *icmp-type-id* is the number of an ICMP reason type.
- *protocol* is an IPv6 protocol number.
- *size* is a number from 0 to 65535.

**Description** This command adds an entry to an IPv6 traffic filter or priority filter. The exact pattern within that entry should not already exist in the filter.

The **filter** parameter specifies the number of the filter to which the entry is to be added. Filters with numbers from 0 to 99 are treated as traffic filters, and use the **action** parameter to specify the action to take with a packet that matches the entry. Filters with numbers from 200 to 299 are treated as priority filters, and use the **priority** parameter to specify the priority to assign to a packet that matches the entry. An interface may have a maximum of one traffic filter and one priority filter, but the same traffic or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the interface, whereas priority filters are applied to packets as they are transmitted.

The **source** parameter specifies the source IP address, in IPv6 notation, for the entry. A prefix length can be specified using slash notation (such as 3FFE::0/16). The prefix is used to determine the portion of the source IPv6 address in the IPv6 packet that is significant for comparison with this entry. The default prefix length is 128.

The **action** parameter specifies, for traffic filters, the action to take when the entry is matched. If **include** is specified, the IP packet is processed and forwarded. If **exclude** is specified, the IP packet is discarded. The **action** and

**priority** parameters are mutually exclusive—only one may be specified. The default is **include**.

The **destination** parameter specifies the destination IP address, in IPv6 notation, for the entry. A prefix length can be specified using slash notation (such as 3FFE::0/16). The prefix is used to determine the portion of the destination IPv6 address in the IPv6 packet that is significant for comparison with this entry. The default prefix length is 128.

The **dport** parameter specifies the port to check against the destination port for this entry, as the recognised name of a well-known UDP or TCP port, a decimal value from 0 to 65535, or a range of numbers formatted as *[low]:[high]*. If *low* is omitted, 0 is assumed. If *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter must also be specified, and must be either **tcp** or **udp**. The default is **any**. The following table lists well-known TCP/UDP ports.

Name	Port	Protocol	Description
ANY	-	-	Any port
BOOTPC	68	UDP	Bootstrap Protocol Client
BOOTPS	67	UDP	Bootstrap Protocol Server
DOMAIN	53	TCP/UDP	Domain Name Server
FINGER	79	TCP	Finger
FTP	21	TCP	File Transfer [Control]
FTPDATA	20	TCP	File Transfer [Default Data]
GOPHER	70	TCP	Gopher
HOSTNAME	101	TCP/UDP	NIC Host Name Server
IPX	213	TCP/UDP	IPX
KERBEROS	88	UDP	Kerberos
LOGIN	49	UDP	Login Host Protocol
MSGICP	29	TCP/UDP	MSG ICP
NAMESERVER	42	UDP	Host Name Server
NEWS	144	TCP	NewS
NNTP	119	TCP	Network News Transfer Protocol
NTP	123	TCP	Network Time Protocol
RTELNET	107	TCP/UDP	Remote Telnet Service
SFTP	115	TCP/UDP	Simple File Transfer Protocol
SMTP	25	TCP	Simple Mail Transfer
SNMP	161	UDP	SNMP
SNMPTRAP	162	UDP	SNMPTRAP
SYSTAT	11	TCP	Active Users
TELNET	23	TCP	Telnet
TFTP	69	UDP	Trivial File Transfer
TIME	37	TCP/UDP	Time
UUCP	540	TCP	uucpd
UUCPRLOGIN	541	TCP/UDP	uucp-rlogin
XNSTIME	52	TCP/UDP	XNS Time Protocol

The **entry** parameter specifies the entry number that this new entry occupies in the filter. Existing entries with the same or higher entry numbers are pushed down the filter. The default is to add the new entry to the end of the filter.

The **icmptype** and **icmpcode** parameters specify the ICMP message type and ICMP message reason code to match against the ICMP type and code fields in an ICMP packet. The **icmptype** parameter specifies the ICMP message type to match as a decimal value from 0 to 255, or the recognised name of an ICMP type. A packet is matched against type and code when the **protocol** parameter is set to ICMP. The default is **any**.

The following table lists predefined ICMP code names and values.

ICMP Code Name	Code Value	Applies to ICMP Type Value	Command Line Syntax
ICMPv6_ANY	-	-	any
ICMPv6_NO_ROUTE_TO_DESTINATION	0	1	noroutetodest
ICMPv6_COMMUNICATION_PROHIBITED	1	1	commsprohibited
ICMPv6_SCOPE_MISMATCH	2	1	scopemismatch
ICMPv6_ADDRESS_UNREACHABLE	3	1	addrunreachable
ICMPv6_PORT_UNREACHABLE	4	1	portunreachable
ICMPv6_HOP_LIMIT_EXCEEDED	0	3	hoplimiitexcd
ICMPv6_REASSEMBLY_TIME_EXCEEDED	1	3	reasmbtimeexc
ICMPv6_ERRONEOUS_HEADER_FIELD	0	4	erroneousheader
ICMPv6_UNRECOGNISED_NEXT_HEADER	1	4	urcnxthead
ICMPv6_UNRECOGNISED_OPTION	2	4	urcoption

The following table lists predefined ICMP type names and values.

ICMP Type Name	Type Value	ICMP Codes Supported	Command Line Syntax
ICMPv6_ANY	-	-	any
ICMPv6_DESTINATION_UNREACHABLE	1	Yes	destunreach
ICMPv6_PACKET_TOO_BIG	2	Yes	pkttoobig
ICMPv6_TIME_EXCEEDED	3	Yes	timeexceeded
ICMPv6_PARAMETER_PROBLEM	4	Yes	paramprob
ICMPv6_ECHO_REQUEST	128	No	echoqrq
ICMPv6_ECHO_REPLY	129	No	echorp
ICMPv6_MULTICAST_LISTENER_QUERY	130	No	mlquery
ICMPv6_MULTICAST_LISTENER_REPORT	131	No	mlrep
ICMPv6_MULTICAST_LISTENER_DONE	132	No	mldone
ICMPv6_ROUTER_SOLICIT	133	No	rtsolicit
ICMPv6_ROUTER_ADVERT	134	No	rtadvert
ICMPv6_NEIGHBOR_SOLICIT	135	No	nbrsolicit
ICMPv6_NEIGHBOR_ADVERT	136	No	nbradvert
ICMPv6_REDIRECT	137	No	redirect
ICMPv6_ROUTER_RENUMBERING	138	No	rtrenumber

The **log** parameter specifies whether matches to a filter entry result in a log message being sent to the switch's logging facility, and the content of the log messages. This parameter enables logging of the IP packet filtering process down to the level of an individual filter entry.

If you specify...	Then...
a number from 4 to 1950	the first 4 to 1950 octets of the data portion of TCP, UDP and ICMP packets or the first 4 to 1950 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP. The filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are also logged with a message type/subtype of IPFIL/PASS (for entries with an INCLUDE action) or IPFIL/FAIL (for entries with an EXCLUDE action).
dump	the filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for entries with an INCLUDE action) or IPFIL/FAIL (for entries with an EXCLUDE action). In addition, the first 40 octets of the data portion of TCP, UDP and ICMP packets, or the first 40 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.
header	the filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/ PASS (for entries with an INCLUDE action) or IPFIL/FAIL (for entries with an EXCLUDE action).
none (default)	matches to the filter entry are not logged.

The **options** parameter specifies the presence or absence of any *time-length-variable* (TLV) encoded "Options" to check against for this entry. TLV-encoded options can be found in the Hop-by-Hop and Destination Options extension headers. If **yes** is specified, the entry matches IP packets with any extension header TLV options set. If **no** is specified, the entry matches IP packets without any extension header TLV options set. The default is **no**.

The **priority** parameter specifies, for priority filters, the priority to apply to forwarding packets when the entry is matched. A low value (**p0**) assigns a high priority to the packet. A high value (**p7**) assigns a low priority to the packet. The priority number is employed during forwarding (transmission). The default is **p7**. The **action** and **priority** parameters are mutually exclusive—only one may be specified.

The **protocol** parameter specifies a protocol to check against the protocol for this entry, as a decimal value from 0 to 65534, or the recognised name of an IP protocol type. If either the **sport** or **dport** parameters are used, **protocol** must be defined as **tcp**, **udp** or **any**. Specifying **tcp** or **udp** filters packets from companion protocols, such as ICMP and OSPF, that do not use TCP or UDP as a transport mechanism. The default is **any**.

The **session** parameter specifies the type of TCP packet to match and is used as a basis for packet filtering when the **protocol** parameter specifies **tcp**. If **start** is specified, the entry matches TCP packets with the SYN bit set and the ACK bit clear. If **established** is specified, the entry matches TCP packets with either the SYN bit clear or the ACK bit set. If **any** is specified, the entry matches any TCP packet. The default is **any**.

The **size** parameter specifies the maximum unassembled size to match against, for each IP packet or fragment. If the fragment's offset plus size is greater than the value specified, the fragment is discarded. The default is **any**, which indicates that size is not required as a matching category.

The **sport** parameter specifies the port to check against the source port for this entry, as the recognised name of a well-known UDP or TCP port, a decimal value from 0 to 65535, or a range of numbers in the form *[low]:[high]*. If *low* is omitted, 0 is assumed. If *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter must also be specified, and must be either **tcp** or **udp**. The default is **any**.

**Examples** To add a filter to block all Telnet traffic, from IP address 3FFE::3 to any IP address whose first 64 contiguous bits match 3FFE::4, and log the header details, use the command:

```
add ipv6 fi=2 so=3ffe::3/128 des=3ffe::4/64 si=ANY prot=tcp
ac=excl sp=any dp=telnet log=h
```

**Related Commands** [add ipv6 interface](#)  
[delete ipv6 filter](#)  
[set ipv6 filter](#)



## add ipv6 host

---

**Syntax** `ADD IPV6 HSt=name IPaddress=ipv6add [INTerface=interface]`

where:

- *name* is a character string 1 to 60 characters long. Valid characters are any printable characters. If the string contains spaces it must be in double quotes.
- *ipv6add* is a valid IPv6 address.
- *interface* is a valid interface.

**Description** This command adds a user-defined name for an IPv6 host to the switch's host name table. The host name table makes it easier to Telnet to commonly accessed hosts, by allowing the user to enter a shorter, easier to remember name for the host rather than the host's full IPv6 address or domain name.

The **host** parameter specifies the user-defined name for the IPv6 host. A host with the same name must not already exist in the host name table. When a host name is specified in the **telnet** command (see the [telnet command on page 62-33 of Chapter 62, Terminal Server](#)), the entire name is used to match a name in the host name table. All characters are used in the comparison, including non-alphabetic characters when they are present. The **host** parameter is not case-sensitive, so the names *MyName* and *myname* are identical.

The **ipaddress** parameter specifies the IPv6 address of the host.

The **interface** parameter specifies the interface, on the switch from which the Telnet request originates, that is then connected to the required destination interface. Telnetting to a link-local address requires interface information as well as the address because a single link-local address can belong to several interfaces. The **interface** parameter is mandatory only if a user is adding a host entry that uses a link-local address. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To add a new host called *foobar* with an IPv6 address of 3FFE::1, use the command:

```
add ipv6 ho=foobar ip=3ffe::1
```

**Related Commands** [delete ipv6 host](#)  
[show ipv6 host](#)  
[show ipv6 interface](#)

## add ipv6 interface

**Syntax** `ADD IPV6 INTerface=interface  
 IPaddress={ipv6add/prefix-length|DHCP|DHCPTemp|PD}  
 [APPINTerface=app-interface[,...]] [DECrement={YES|NO|  
 ON|OFF|ENAbLe|DISAbLe}}] [FILter=0..99] [HINT=ipv6add/  
prefix-length] [KEY=key-id] [PREferred=1..4294967295|  
 INFinite] [PRIorityfilter=200..299] [PUBlish={YES|NO|  
 ON|OFF|ENAbLe|DISAbLe}}] [TYPe={ANYcast|UNICast}}]  
 [VALId=1..4294967295|INFinite]`

where:

- *interface* is a valid interface.
- *ipv6add* is a valid IPv6 address, with its prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is an integer from 1 to 128.
- *app-interface* is a valid interface, optionally followed by a slash and an integer from 1 to 65535 (e.g. ppp1/2).
- *key-id* is a number from 0 to 65535.

**Description** This command adds an IPv6 address to an interface, with a specified lifetime. If no lifetime is specified then the lifetime is infinite. If the interface has not already been created, this command creates it. You can add up to 32 IPv6 addresses to an interface.

The **interface** parameter is the interface to add the IPv6 address to. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **ipaddress** parameter specifies one of the following:

- the IPv6 address to assign to the interface, in IPv6 notation. A prefix length must be specified, using slash notation (e.g. 3FFE::0/16)
- that the interface request a permanent IPv6 address from a DHCP6 server (the **dhcp** option)
- that the interface request a temporary IPv6 address from a DHCP6 server (the **dhcptemp** option). Using temporary addresses increases security
- that the interface request a prefix from a DHCP6 server in order for the switch to apply the prefix to other IPv6 interfaces (the **pd** option). The **appint** parameter must also be specified and lists interfaces to which the prefix is assigned.

An IPv6 interface that is used as a DHCP server must not use DHCP to configure its IPv6 address. To stop the switch from using a DHCP6 server to configure its IP address, use the **delete dhcp6 interface** command. To remove an IP address from an interface, use the **delete ipv6 interface** command.

The **appint** parameter is required when the interface obtains a prefix by prefix delegation from DHCP6 in order to assign the prefix to other interfaces. It

specifies the interface or comma-separated interfaces to which the received prefix is assigned. **appint** is required if the **ipaddress** parameter is set to **pd**. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

The interface is optionally followed by a value that determines the next 16 bits for the prefix assigned to this interface, if the delegated prefix has a prefix length of less than 64 bits (for example, ppp1/2). If a value is not specified, the switch assigns a value of 1 to the first **appint**, 2 to the second **appint**, and so on.

The **decrement** parameter specifies whether to decrement the time counters for valid and preferred IPv6 address lifetimes, and published prefix lifetimes. IPv6 addresses with valid and preferred lifetimes set to infinite are not decremented. The default is **no**.

The **filter** parameter is the number of the traffic filter that is to be applied to this interface.

The **hint** parameter is used with DHCP6 requests. It specifies one of:

- a particular IPv6 address to request, if **ipaddress** is **dhcp** or **dhcptemp**
- a particular prefix and/or prefix length to request if **ipaddress** is **pd**

Requesting a particular address or prefix is no guarantee of receiving it because the server may already have assigned it to another client. To guarantee that a client always receives a particular address or prefix, create a static entry by using the command [add dhcp6 range in Chapter 35, Dynamic Host Configuration Protocol for IPv6 \(DHCP6\)](#).

The **key** parameter is used to configure authentication of DHCP6 message exchanges, and specifies the ID number of the authentication key.

The **preferred** parameter specifies the time in seconds for which this IPv6 address is to be the preferred address for this interface. The default is 604800 seconds (7 days). The value of this parameter cannot be greater than that of the **valid** parameter.

The **priorityfilter** parameter is the number of the priority filter that is to be attached to this interface.

The **publish** parameter determines whether to include the prefix in router advertisement packets. The default is **no**. To comply with Section 6.2.1 of RFC 2461, the switch does not generate router advertisements by default. To enable the switch to generate router advertisements that include this switch's address, you must specify **publish=yes**, and also enable advertisements using the **enable ipv6 advertise** command. If the switch does not generate router advertisements, the router discovery and address autoconfiguration processes in the network may not work as expected.

The **type** parameter allows an anycast address to be distinguished from a unicast address. The default is **unicast**.

The **valid** parameter is the time in seconds that the IPv6 address exists on the interface. After this time, the IPv6 address is deleted. The default is 2592000 seconds (30 days). The time value must be the same as or greater than that of the **preferred** parameter.

**Examples** To add an IPv6 address of 3FFE::1/32 to vlan1, use the command:

```
add ipv6 ip=3ffe::/32 int=vlan1
```

To request an IPv6 prefix with a prefix length of 48 bits from vlan1 to be delegated on vlan2, use the command:

```
add ipv6 int=vlan1 ip=pd appint=vlan2  
hi=::/48
```

**Related Commands**

- [create ipv6 interface](#)
- [delete ipv6 interface](#)
- [destroy ipv6 interface](#)
- [enable ipv6 advertise](#)
- [set ipv6 interface](#)
- [show ipv6](#)
- [show ipv6 interface](#)

## add ipv6 nd

---

**Syntax** ADD IPV6 ND=*ipv6add* INTerface=*interface* ETHernet=*macadd*  
[ISRouter={YES|NO}] [Port=*port-number*]

where:

- *ipv6add* is a valid IPv6 address.
- *interface* is a valid interface.
- *macadd* is a MAC address in hexadecimal notation.
- *port-number* is the physical switch port number. Port numbers start at 1 and end at *m*, where *m* is the highest numbered Ethernet switch port, including uplink ports.

**Description** This command manually adds a neighbour to the neighbour cache.

The **nd** (neighbour discovery) parameter specifies the IPv6 address of the neighbour.

The **interface** parameter specifies the interface on which the switch receives data from the neighbour. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **ethernet** parameter specifies the MAC address (the Ethernet link-layer address) of the neighbour.

The **isrouter** parameter specifies whether the added neighbour is a router. The default is **no**.

The **port** parameter specifies the physical switch port number in a VLAN. If the **interface** parameter specifies a VLAN interface, the **port** parameter is required. Otherwise the **port** parameter is not valid.

**Examples** To manually add a host as a neighbour, with the MAC address aa-bb-cc-dd-11-22 and the IPv6 address 3ffe::1, to port 5 on the vlan1 interface, use the command:

```
add ipv6 nd=3ffe::1 int=vlan1 port=5 eth=aa-bb-cc-dd-11-22
```

**Related Commands** [delete ipv6 nd](#)  
[reset ipv6 ndcache](#)  
[show ipv6 interface](#)  
[show ipv6 ndcache](#)

## add ipv6 prefix

---

**Syntax** `ADD IPV6 PREFIX=ipv6add/prefix-length INTerface=interface  
[AUTOonomous={YES|NO}] [ONlink={YES|NO}]  
[PREferred=1..4294967295|INFinite]  
[VALid=1..4294967295|INFinite]`

where:

- *ipv6add* is a valid IPv6 prefix, with the prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is the number between 1 and 128.
- *interface* is a valid interface.

**Description** This command specifies a prefix for the switch to include in its advertisement messages for the specified interface, without adding the prefix to the interface's IPv6 address. The prefix can assist nodes on the subnet with configuration of their local-link IPv6 addresses. More than one prefix can be included in the router advertisements for a given interface.

If the **publish** parameter is set to **yes** for an interface using the [add ipv6 interface command on page 34-42](#) or the [set ipv6 interface command on page 34-68](#), the prefix specified by that command is advertised as well as the prefix specified here.

The **prefix** parameter specifies the IPv6 prefix (e.g. 3ffe::/64) to advertise for this interface.

The **interface** parameter is the interface to add the IPv6 address to. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **autonomous** parameter specifies whether the prefix is for the same autonomous system (for example, the Internet or an intranet). If it is, hosts use this prefix to configure their addresses over this autonomous system. The default is **yes**.

The **onlink** parameter specifies whether hosts should use this prefix to configure their addresses on the local link. The default is **yes**.

The **preferred** parameter specifies the time in seconds for which the prefix is preferred over other prefixes. The prefix no longer appears in router advertisements after this period. The default is 604800 seconds (7 days).

The **valid** parameter specifies the time in seconds for which the prefix is valid. The prefix is deleted after this period. The default is 2592000 seconds (30 days).

**Examples** To include the prefix 3ffe::/64 in router advertisements sent over the vlan1 interface, use the command:

```
add ipv6 prefix=3ffe::/64 int=vlan1
```

**Related Commands**    [delete ipv6 prefix](#)  
[set ipv6 prefix](#)  
[show ipv6 interface](#)  
[show ipv6 ndcache](#)

## add ipv6 rip

---

**Syntax**    `ADD IPV6 RIP INTerface=interface POISONreverse={ON|OFF|True|False}`

**Description**    This command enables the switch to listen for RIPv6 packets on the specified interface.

The **interface** parameter is the physical interface to listen for RIP packets on. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **poisonreverse** parameter specifies whether poison reverse is enabled on the interface. To enable poison reverse, specify **on** or **true**. To disable poison reverse, specify **off** or **false**. Poison reverse addresses the problem of slow convergence on RIPv6 routes. If one device in a network goes down, it can take a long time for the devices to recognise that routes through the crashed device are no longer available. If poison reverse is enabled when a device goes down, the devices next to it continues to advertise the route but with a cost of 16. This cost indicates that the route is unavailable.

**Examples**    To enable listening for RIP packets on vlan1, use the command:

```
add ipv6 rip int=vlan1 poi=on
```

**Related Commands**    [delete ipv6 rip](#)  
[disable ipv6 rip](#)  
[enable ipv6 rip](#)  
[show ipv6 interface](#)  
[show ipv6 rip](#)

## add ipv6 route

---

**Syntax** ADD IPV6 ROUTe=*ipv6add/prefix-length* INTerface=*interface*  
[NEXThop=*ipv6add*] [METric=1..16] [PREference=0..65535]

where:

- *prefix-length* is an integer from 1 to 128.
- *ipv6add* is a valid IPv6 address, with its prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *interface* is a valid interface.

**Description** This command adds a static route to the IPv6 route table. The static route must not already exist. However, if the route exists as a dynamic (e.g. RIP-derived) route, the static route may still be added. To specify that the destination node is on the network to which traffic is being directed, the **nexthop** parameter should be 0:0:0:0:0:0:0:0 (or ::).

Static routes can be used to define default routes to external routers or networks. A default route is one with a network address of 0:0:0:0:0:0:0:0 (or ::). If the switch receives data and cannot find a route for it, the data is sent to the default route. To define a default route, set **route** to :: and set **nexthop** to point to the network (router) to which default packets are to be directed.

The **route** parameter is the IPv6 address of a host or network to which packets are to be routed. Slash notation can be used on this parameter.

The **interface** parameter is the physical interface over which the switch forwards packets. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **nexthop** parameter is the IPv6 address of the next router along the path to the destination, and is required for all interfaces except static tunnel virtual (VIRT) interfaces and PPP. For routes through 6-to-4 tunnel VIRT interfaces, it is required and is the IPv6 address of the router at the other end of the tunnel.

The **metric** parameter specifies the cost of the route. The cost is used in RIP entries to determine the best path to a node. The default is 1.

The **preference** parameter specifies the preference for the route. When more than one route in the route table matches the destination address in an IPv6 packet, the route with the lowest preference value is used to route the packet. If two or more candidate routes have the same preference, the route with the longest prefix is used. Interface routes have a preference of 0 and RIP routes have a preference of 100. The default preference for static routes other than 0:0:0:0:0:0:0:0 is 60. The default for the default static route 0:0:0:0:0:0:0:0 is 360.

**Examples** To add a route to the network 3FFE::/16, using a gateway with the IPv6 address 4FFE::1 that is connected to the switch on vlan1, use the command:

```
add ipv6 ro=3ffe::/16 next=4ffe::1 int=vlan1
```



**Related Commands** [delete ipv6 route](#)  
[show ipv6](#)  
[show ipv6 interface](#)  
[show ipv6 route](#)

## add ipv6 tunnel

---

**Syntax** `ADD IPV6 TUNnel LOcal=ipv4add TArget=ipv4add  
[INTerface=tunnel-interface] [IPaddress=ipv6add]`

where:

- *ipv4add* is a valid IPv4 address in dotted decimal notation.
- *tunnel-interface* is a virtual interface of the form *virt**n*, where *n* is an integer between 0 and 255.
- *ipv6add* is a valid IPv6 address (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).

**Description** This command creates a static tunnel between two IPv6 networks, to link them over an IPv4 network. A tunnel must be created on both of the IPv6 switches or routers that link the two networks, and a route that uses the tunnel must be added on each end.

The **local** parameter is the IPv4 address of the interface on the switch that is used in the tunnel.

The **target** parameter is the IPv4 address of the last node in the tunnel. The end node must be capable of forwarding an IPv6 packet. A tunnel must also be configured on the remote router or switch.

The **interface** parameter specifies a virtual tunnel interface for this tunnel (for example, *virt5*). If this parameter is not specified, the tunnel is created on the first available virtual interface.

The **ipaddress** parameter specifies the IPv6 address of the tunnel, if an address is being added to the tunnel manually. If this parameter is not specified, the switch assigns a link-local IPv6 address to the tunnel (see [“Static tunnelling” on page 34-21](#)).

**Examples** To add a tunnel between two switches A and B, use the following commands on switch A:

```
add ip int=vlan1 ip=192.168.1.1
add ipv6 tun lo=192.168.1.1 ta=192.168.1.2
```

Add the following commands on switch B:

```
add ip int=vlan1 ip=192.168.1.2
add ipv6 tun lo=192.168.1.2 ta=192.168.1.1
```

**Related Commands** [add ipv6 route](#)  
[delete ipv6 tunnel](#)  
[show ipv6 tunnel](#)

## create ipv6 interface

---

**Syntax** `CREate IPV6 INTerface=interface [DUPtrans=1..16]  
[METric=1..16] [RETRans=0..4294967295]`

**Description** This command creates an IPv6 Ethernet interface and uses stateless address autoconfiguration to assign it a link-local address. The address is formed from the link-layer address and the prefix FE80::. To create a PPP interface and assign it an IPv6 address, use the **add ipv6 interface** command. To create a VIRT interface, use the **add ipv6 tunnel** command.

You can create up to 1023 IPv6 interfaces.

The **interface** parameter specifies the interface that is to be created for IPv6. Routing using this interface is then possible. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **duptrans** parameter is the DupAddrDetectTransmits value. This is the number of Neighbour Solicitation messages that the switch sends while performing Duplicate Address Detection on a tentative address. The default is 1.

The **metric** parameter specifies the cost to RIPv6 for crossing the logical interface. This parameter is allowed only on link-local interfaces. Therefore, setting this parameter also sets metrics for all logical interfaces over the same IPv6 interface to the same value. The default is 1. For more information about RIPv6, see [“IPv6 Routing” on page 34-11](#).

The **retrans** parameter is the AdvRetransTimer, in milliseconds. This is the interval between repeats of each Router Advertisement message sent by the switch. The value you enter is rounded up to the nearest 100 milliseconds (for example, 301 becomes 400). The default is 0, which indicates that this timer is not specified by this switch.

**Examples** To create an IPv6 interface, having a metric of 6, use the command:

```
cre ipv6 int=vlan1 metric=6
```

**Related Commands**

- [add ipv6 interface](#)
- [add ipv6 tunnel](#)
- [delete ipv6 interface](#)
- [destroy ipv6 interface](#)
- [set ipv6 interface](#)
- [set ipv6 nd](#)
- [show ipv6](#)
- [show ipv6 interface](#)

---

## delete ipv6 6to4

---

**Syntax** `DELEte IPV6 6TO4 IP=ipv4add`

where *ipv4add* is a valid IPv4 address in dotted decimal notation

**Description** This command removes the specified 6-to-4 interface from the switch.

The **ip** parameter specifies the IPv4 address of the 6-to-4 interface that is to be removed.

**Related Commands** [add ipv6 6to4](#)  
[add ipv6 tunnel](#)  
[show ipv6 tunnel](#)

---

## delete ipv6 filter

---

**Syntax** `DELEte IPV6 FILter=filter-id ENTRy={entry-number|ALL}`

where:

- *filter-id* is a number from 0 to 99 or 200 to 299.
- *entry-number* is the position of this entry in the filter.

**Description** This command deletes an entry from an IP traffic filter or priority filter. The entry must already exist in the filter.

The **filter** parameter specifies the number of the filter from which the entry is to be deleted. Filters with numbers from 0 to 99 are traffic filters and filters with numbers from 200 to 299 are priority filters.

The **entry** parameter specifies the entry number in the filter that is to be deleted. Existing entries with the same or higher entry numbers are pushed up the filter to occupy the vacant entry. If **all** is specified, the filter is deleted.

**Examples** To delete entry 2 from filter 2, use the command:

```
del ipv6 fil=2 ent=2
```

To delete all entries from filter 2, use the command:

```
del ipv6 fil=2 ent=all
```

**Related Commands** [add ipv6 filter](#)  
[set ipv6 filter](#)

## delete ipv6 host

---

**Syntax** `DELEte IPV6 HOsT=name`

where *name* is a character string 1 to 60 characters long

**Description** This command removes a host name from the host name table.

The **host** parameter is a name that has been associated with an IPv6 address by using the **add ipv6 host** command.

**Examples** To remove the host entry *foobar*, use the command:

```
del ipv6 ho=foobar
```

**Related Commands** [add ipv6 host](#)  
[show ipv6 host](#)

## delete ipv6 interface

---

**Syntax** `DELEte IPV6 INTerface=interface IPaddress=ipv6add`

where:

- *interface* is a valid interface.
- *ipv6add* is a valid IPv6 address.

**Description** This command removes an IPv6 address from an interface. To stop an interface from using a DHCP6 server to configure its IP address, use the **delete dhcp6 interface** command instead.

The **interface** parameter specifies the interface on the switch to remove the IPv6 address from. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **ipaddress** parameter specifies the IPv6 address to remove from the interface.

**Examples** To remove the address 3FFE::1 from the vlan1 interface, use the command:

```
del ipv6 int=vlan1 ip=3ffe::1
```

**Related Commands** [add ipv6 interface](#)  
[create ipv6 interface](#)  
[destroy ipv6 interface](#)

```
set ipv6 interface  
show ipv6  
show ipv6 interface
```

## delete ipv6 nd

---

**Syntax** DELEte IPV6 ND=*ipv6add* INTerface=*interface*

where:

- *ipv6add* is a valid IPv6 address.
- *interface* is a valid interface.

**Description** This command deletes a dynamically learned neighbour from the neighbour cache. It also removes a statically configured neighbour.

The **nd** parameter specifies the IPv6 address of the neighbour.

The **interface** parameter specifies the interface on which the switch receives data from the neighbour. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To specify that the host with IPv6 address 3ffe::1 is no longer a neighbour on the vlan1 interface, use the command:

```
del ipv6 nd=3ffe::1 int=vlan1
```

**Related Commands**

- [add ipv6 nd](#)
- [reset ipv6 ndcache](#)
- [show ipv6 interface](#)
- [show ipv6 ndcache](#)

## delete ipv6 prefix

---

**Syntax** `DELEte IPV6 PREFIX=ipv6add/prefix-length  
INTErface=interface`

where:

- *ipv6add* is a valid IPv6 prefix, with the prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is the number between 1 and 128.
- *interface* is a valid interface.

**Description** This command removes the IPv6 prefix from the prefix list. This prevents this prefix from being advertised via Router Advertisements.

The **interface** parameter specifies the interface on the switch to remove the IPv6 prefix from. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To stop the vlan1 interface from advertising the prefix 3ffe::/64, use the command:

```
del ipv6 prefix=3ffe::/64 int=vlan1
```

**Related Commands**

- [add ipv6 prefix](#)
- [set ipv6 prefix](#)
- [show ipv6 interface](#)
- [show ipv6 ndcache](#)

## delete ipv6 rip

---

**Syntax** DELEte IPV6 RIP INTErface=*interface*

where *interface* is a valid interface

**Description** This command stops an interface from sending or receiving RIP packets.

The **interface** parameter specifies the physical interface that the switch was sending and receiving RIP packets on. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To stop the vlan1 interface from sending or receiving RIP packets, use the command:

```
del ipv6 rip int=vlan1
```

**Related Commands**

- [add ipv6 rip](#)
- [disable ipv6 rip](#)
- [enable ipv6 rip](#)
- [show ipv6 interface](#)



---

## delete ipv6 route

---

**Syntax** `DELEte IPV6 ROUte=ipv6add INTerface=interface  
NEXThop=ipv6add`

where:

- *ipv6add* is a valid IPv6 address.
- *interface* is a valid interface.

**Description** This command removes a route from the IPv6 route table.

The **route** parameter specifies the IPv6 address of a host or network to which packets were being routed.

The **interface** parameter specifies the physical interface out of which the switch was forwarding packets. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **nexthop** parameter specifies the IPv6 address of the router that was next on the path to the destination.

**Examples** To delete the route to 3FFE:: with a next hop of 3FFE::1 on vlan1, use the command:

```
del ipv6 rou=3ffe:: nest=3ffe::1 int=vlan1
```

**Related Commands** [add ipv6 route](#)  
[show ipv6 interface](#)  
[show ipv6 route](#)

## delete ipv6 tunnel

---

**Syntax** `DELEte IPV6 TUNnel=ipv6add INTErface=tunnel-interface`

where:

- *ipv6add* is the first link-local address on the tunnel interface (the address that the switch assigned to the tunnel when it was created).
- *tunnel-interface* is a virtual interface of the form *virt**n*, where *n* is an integer between 0 and 255.

**Description** This command removes an IPv6 tunnel. The tunnel's address and interface can be seen by using the **show ipv6 tunnel** command.

The **tunnel** parameter specifies the IPv6 address of the tunnel to be removed. The tunnel's IPv6 address is the address that was assigned to the tunnel interface when the tunnel was created.

The **interface** parameter specifies the virtual interface from which the tunnel is to be removed.

If the PIM6 interface and the MLF interface are active, these must be deleted before deleting the IPv6 tunnel. An error message is displayed if you fail to do this.

**Examples** To delete the tunnel with an IPv6 address of fe80::c0a8:0102 on *virt1*, use the command:

```
del ipv6 tun=fe80::c0a8:0102 int=virt1
```

**Related Commands**

- [add ipv6 tunnel](#)
- [delete pim6 interface](#)
- [disable ipv6 mld](#)
- [show ipv6 tunnel](#)

---

## destroy ipv6 interface

---

**Syntax** DESTroy IPV6 INTerface=*interface*

where *interface* is a valid interface name

**Description** This command removes an IPv6 interface. Note that destroying an interface deletes all routes attached to the interface.

The **interface** parameter specifies the physical interface that is to be removed from IPv6. Routing using this interface is no longer possible. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To remove the vlan1 interface, use the command:

```
dest ipv6 int=vlan1
```

**Related Commands** [add ipv6 interface](#)  
[create ipv6 interface](#)  
[delete ipv6 interface](#)  
[set ipv6 interface](#)  
[show ipv6 interface](#)

---

## disable ipv6

---

**Syntax** DISable IPV6

**Description** This command disables the IPv6 module. Any dynamic configuration, such as address autoconfiguration, is preserved between disabling and re-enabling the IPv6 module. The IPv6 module is disabled by default.

**Examples** To disable the IPv6 module, use the command:

```
dis ipv6
```

**Related Commands** [enable ipv6](#)  
[show ipv6](#)

## disable ipv6 advertise

---

**Syntax** `DISable IPV6 ADVERTISE [INTERface=interface]`

where *interface* is a valid interface

**Description** This command disables sending of router advertisement packets. Router advertisements are disabled by default.

If the **interface** parameter is specified, sending router advertisements is enabled on that interface. Otherwise, sending router advertisement packets is enabled for all IPv6 interfaces. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To disable router advertisements on all interfaces, use the command:

```
dis ipv6 adv
```

**Related Commands** [enable ipv6 advertise](#)  
[show ipv6](#)  
[show ipv6 interface](#)

## disable ipv6 debug

---

**Syntax** `DISable IPV6 DEBug`

**Description** This command disables debugging of IPv6 packets. Debugging is disabled by default.

**Examples** To disable IPv6 debugging, use the command:

```
dis ipv6 deb
```

**Related Commands** [enable ipv6 debug](#)  
[show ipv6](#)

---

## disable ipv6 mtudiscovery

---

**Syntax** DISable IPV6 MTUdiscovery

**Description** This command disables Path MTU Discovery. The switch does not automatically increase the Path MTU and ignores all ICMPv6 Packet Too Big Messages. Path MTU Discovery is disabled by default.

**Examples** To disable Path MTU Discovery, use the command:

```
dis ipv6 mtu
```

**Related Commands** [enable ipv6 mtudiscovery](#)

---

## disable ipv6 rip

---

**Syntax** DISable IPV6 RIP

**Description** This command disables routing of IPv6 packets using RIPv6. RIP is disabled by default.

**Examples** To disable the sending and receiving of RIP packets, use the command:

```
dis ipv6 rip
```

**Related Commands** [add ipv6 rip](#)  
[delete ipv6 rip](#)  
[enable ipv6 rip](#)  
[show ipv6 rip](#)

---

## enable ipv6

---

**Syntax** ENAbble IPV6

**Description** This command enables the IPv6 module. Any dynamic configuration is preserved between disabling and re-enabling the IPv6 module. The IPv6 module is disabled by default.

**Examples** To enable IPv6, use the command:

```
ena ipv6
```

**Related Commands** [disable ipv6](#)  
[show ipv6](#)

## enable ipv6 advertise

---

ENable IPV6 ADVERTISE [INTERface=*interface*]

where *interface* is a valid interface

**Description** This command enables sending of router advertisement packets. To comply with Section 6.2.1 of RFC 2461, *IPv6 Neighbour Discovery*, the switch does not generate router advertisements by default.

As well as using this command to enable advertisements, for each IPv6 interface you want to have advertised, also specify that the interface's prefix should be included in advertisements. Use one of the commands:

```
add ipv6 interface=interface ipaddress={ipv6add/prefix
length|dhcp|dhcptemp|pd} publish=yes
set ipv6 interface=interface publish=yes
```

If the switch does not generate router advertisements, the router discovery and address autoconfiguration processes in the network may not work as expected.

If the **interface** parameter is specified, sending router advertisements is enabled on that interface. Otherwise, sending router advertisement packets is enabled for all IPv6 interfaces. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To enable router advertisements on all interfaces, use the command:

```
ena ipv6 adv
```

**Related Commands** [disable ipv6 advertise](#)  
[show ipv6](#)  
[show ipv6 interface](#)

## enable ipv6 debug

---

**Syntax** ENable IPV6 DEBug

**Description** This command enables debugging of IPv6 packets. Debugging is disabled by default.

**Examples** To enable debugging, use the command:

```
ena ipv6 deb
```

**Related Commands** [disable ipv6 debug](#)  
[show ipv6](#)

## enable ipv6 mtudiscovery

---

**Syntax**    ENABle IPV6 MTUdiscovery

**Description**    This command enables Path MTU Discovery. This mechanism allows the switch to determine the actual maximum transmission units (MTU) of nodes on possible paths, and use a path with a higher PMTU than the IPv6 minimum link MTU if such a path exists. Path MTU Discovery is disabled by default.

**Examples**    To enable Path MTU Discovery, use the command:

```
ena ipv6 mtu
```

**Related Commands**    [disable ipv6 mtudiscovery](#)

## enable ipv6 rip

---

**Syntax**    ENABle IPV6 RIP

**Description**    This command enables routing of IPv6 packets using RIPv6. RIPv6 is disabled by default.

**Examples**    To enable the sending and receiving of RIPv6 packets, use the command:

```
ena ipv6 rip
```

**Related Commands**    [add ipv6 rip](#)  
[delete ipv6 rip](#)  
[disable ipv6 rip](#)  
[show ipv6 rip](#)

## reset ipv6 ndcache

---

**Syntax** RESET IPV6 NDcache

**Description** This command flushes the Neighbour Discovery cache of all entries except those manually configured with the **add ipv6 nd** command. Stale ND entries are automatically flushed every 5 minutes.

**Examples** To purge the Neighbour Discovery cache of entries not manually configured, use the command:

```
reset ipv6 nd
```

**Related Commands** [add ipv6 nd](#)  
[show ipv6 ndcache](#)  
[delete ipv6 nd](#)



## set ipv6 filter

**Syntax** SET IPV6 FILTER=*filter-id* ENTRY=*entry-number*  
 SOURCE=*ipv6add/prefix-length* [ACTION={INCLUDE|EXCLUDE} |  
 PRIORITY=P0..P7] [DESTINATION=*ipv6add/prefix-length*]  
 [DPORT={*port-name*|*port-id*|Any}]  
 [ICMPCODE={*icmp-code-name*|*icmp-code-id*|Any}]  
 [ICMPTYPE={*icmp-type-name*|*icmp-type-id*|Any}]  
 [LOG={4..1950|Dump|Header|None}] [OPTIONS={YES|NO|ON|  
 OFF|True|False}] [PROTOCOL={*protocol*|Any|Egp|Icmp|Ospf|  
 Tcp|Udp}] [SESSION={Any|Established|Start}]  
 [SIZE={*size*|Any}] [SPORT={*port-name*|*port-id*|Any}]

where:

- *filter-id* is an integer from 0 to 99 or 200 to 299.
- *entry-number* is the position of this entry in the filter.
- *ipv6add* is a valid IPv6 address, with its prefix length optionally indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is an integer from 1 to 128.
- *port-name* is the predefined name of a TCP or UDP port.
- *port-id* is an IP port number or a range of port numbers in the format [*low*]:[*high*].
- *icmp-code-name* is the predefined name for an ICMP reason code.
- *icmp-code-id* is the number of an ICMP reason code.
- *icmp-type-name* is the predefined name of an ICMP message type.
- *icmp-type-id* is the number of an ICMP message type.
- *protocol* is an IPv6 protocol number.
- *size* is an integer from 0 to 65535.

**Description** This command changes an entry in an IPv6 traffic filter or priority filter.

The **filter** parameter specifies the number of the filter in which the entry is to be changed. Filters with numbers from 0 to 99 are treated as traffic filters, and use the **action** parameter to specify the action to take with a packet that matches the entry. Filters with numbers from 200 to 299 are treated as priority filters, and use the **priority** parameter to specify the priority to assign to a packet that matches the entry. An interface may have a maximum of one traffic filter and one priority filter. The same traffic or priority filter can be assigned to more than one interface. Traffic filters are applied to packets received via the interface, whereas priority filters are applied to packets as they are transmitted.

The **entry** parameter specifies the entry number to be changed.

The **action** parameter specifies, for traffic filters, the action to take when the entry is matched. If **include** is specified, the IP packet is processed and forwarded. If **exclude** is specified, the IP packet is discarded. The **action** and **priority** parameters are mutually exclusive—only one may be specified. The default is **include**.

The **destination** parameter specifies the destination IP address, in IPv6 notation, for the entry. A prefix length can be specified using slash notation (such as 3FFE::0/16). The prefix is used to determine the portion of the

destination IPv6 address in the IPv6 packet that is significant for comparison with this entry. The default prefix length is 128.

The **dport** parameter specifies the destination port to check against for this entry as the recognised name of a well-known UDP or TCP port, a decimal value from 0 to 65535, or a range of numbers in the form *[low]:[high]*. If *low* is omitted, 0 is assumed. If *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter must also be specified, and must be one of **tcp** or **udp**. The default is **any**.

The **icmpcode** parameter specifies the ICMP message reason code to match against the ICMP code field in an ICMP packet as a decimal value from 0 to 255, or the recognised name of an ICMP reason code. This parameter is the basis for packet matching when the **protocol** parameter is set to **icmp**. The default is **any**.

The **icmptype** parameter specifies the ICMP message type to match against the ICMP type field in an ICMP packet header as a decimal value from 0 to 255 or the recognised name of an ICMP type. This parameter is the basis for packet matching when the **protocol** parameter is set to **icmp**. The default is **any**.

The **log** parameter specifies whether matches to a filter entry result in a log message being sent to the switch's logging facility, and the content of the log messages. This parameter enables logging of the IP packet filtering process down to the level of an individual filter entry.

If you specify...	Then...
a number from 4 to 1950	the first 4 to 1950 octets of the data portion of TCP, UDP and ICMP packets or the first 4 to 1950 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP. The filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are also logged with a message type/subtype of IPFIL/PASS (for entries with an <b>include</b> action) or IPFIL/FAIL (for entries with an <b>exclude</b> action).
dump	the filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for entries with an <b>include</b> action) or IPFIL/FAIL (for entries with an <b>exclude</b> action). In addition, the first 40 octets of the data portion of TCP, UDP and ICMP packets, or the first 40 octets after the IP header of other protocol packets are logged with a message type/subtype of IPFIL/DUMP.
header	the filter number, entry number and IP header information (source and destination IP addresses, protocol, source and destination ports, and size) are logged with a message type/subtype of IPFIL/PASS (for entries with an <b>include</b> action) or IPFIL/FAIL (for entries with an <b>exclude</b> action).
none (default)	matches to the filter entry are not logged.

The **options** parameter specifies the presence or absence of any *time-length-variable* (TLV) encoded "Options" to check against. TLV-encoded options can be found in the Hop-by-Hop and Destination Options extension headers. If **yes** is specified, the entry matches IP packets with any extension header TLV options

set. If **no** is specified, the entry matches IP packets without any extension header TLV options set. The default is **no**.

The **priority** parameter specifies, for priority filters, the priority to apply to forwarding packets when the entry is matched. A low value (**p0**) assigns a high priority to the packet. A high value (**p7**) assigns a low priority to the packet. The priority number is employed during forwarding (transmission). The default is **p7**. The **action** and **priority** parameters are mutually exclusive—only one may be specified.

The **protocol** parameter specifies a protocol to check against as a decimal value from 0 to 65534, or the recognised name of an IP protocol type. If either the **sport** or **dport** parameters are used, **protocol** must be defined as **tcp**, **udp** or **any**. Specifying **tcp** or **udp** filters packets from companion protocols, such as ICMP and OSPF, that do not use TCP or UDP as a transport mechanism. The default is **any**.

The **session** parameter specifies the type of TCP packet to match and is used as a basis for packet filtering when the **protocol** parameter specifies TCP. If **start** is specified, the entry matches TCP packets with the SYN bit set and the ACK bit clear. If **established** is specified, the entry matches TCP packets with either the SYN bit clear or the ACK bit set. If **any** is specified, the entry matches any TCP packet. The default is **any**.

The **size** parameter specifies the maximum unassembled size to match against for each IP packet or fragment. If the fragment's offset plus size is greater than the value specified, the fragment is discarded. The default is **any**, which indicates that size is not required as a matching category.

The **source** parameter specifies the source IP address, in IPv6 notation, for the entry. A prefix length can be specified using slash notation (such as 3FFE::0/16). The prefix is used to determine the portion of the source IPv6 address in the IPv6 packet that is significant for comparison with this entry. The default prefix length is 128.

The **sport** parameter specifies the source port to check against as the recognised name of a well-known UDP or TCP port, a decimal value from 0 to 65535, or a range of numbers in the form *[low]:[high]*. If *low* is omitted, 0 is assumed. If *high* is omitted, the maximum port number is assumed. If a port other than **any** is specified, the **protocol** parameter must also be specified, and must be either **tcp** or **udp**. The default is **any**.

**Examples** To set a filter to monitor and log all Telnet sessions from IP address 3ffe::3, to IP address 3ffe::4, use the command:

```
set ipv6 fil=2 so=3ffe::3/128 des=3ffe::4/64 si=any prot=tcp
ac=incl sp=any dp=23 log=h
```

**Related Commands** [add ipv6 filter](#)  
[delete ipv6 filter](#)

## set ipv6 interface

---

**Syntax** SET IPV6 INTErface=*interface* [FILter=0..99]  
[IPaddress=*ipv6add/prefix-length*] [METric=1..16]  
[PREFerred=1..4294967295|INFInite]  
[PRIorityfilter=200..299] [PUBlish={YES|NO|ON|OFF|True|False}] [VALid=1..4294967295|INFInite]

where:

- *interface* is a valid interface.
- *ipv6add* is a valid IPv6 address, with its prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is an integer from 1 to 128.

**Description** This command modifies values associated with an interface that was created by either the **create ipv6 interface** or **add ipv6 interface** command.

The **interface** parameter specifies the physical interface to set the properties of. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **filter** parameter is the number of the filter that is to be applied to this interface.

The **ipaddress** parameter specifies the IPv6 address to associate with this interface, in IPv6 notation. A prefix length must be specified, using slash notation (e.g. 3FFE::0/16). **ipaddress** is required if the **valid**, **preferred**, or **publish** parameters are specified.

The **metric** parameter specifies the cost to RIPv6 for crossing the logical interface. This parameter is allowed only on link-local interfaces. Therefore, setting this parameter also sets the metrics for all logical interfaces over the same IPv6 interface to the same value. The default is 1. For more information about RIPv6, see [“IPv6 Routing” on page 34-11](#).

The **preferred** parameter specifies the time in seconds for which this IPv6 address is the preferred address for this interface. The default is 604800 seconds (7 days). The value of this parameter cannot be greater than that of the **valid** parameter. The **preferred** parameter can be specified if an IPv6 address is specified by using the **ipaddress** parameter.

The **priorityfilter** parameter is the number of the priority filter attached to this interface.

The **publish** parameter determines whether to include the prefix in router advertisement packets. The default is **no**. To comply with Section 6.2.1 of RFC 2461, the switch does not generate router advertisements by default. To enable the switch to generate router advertisements that include this switch's address, you must specify **publish=yes**, and also enable advertisements using the [enable ipv6 advertise command on page 34-62](#). If the switch does not

generate router advertisements, the router discovery and address autoconfiguration processes in the network may not work as expected. The **publish** parameter can be specified if an IPv6 address is specified by using the **ipaddress** parameter.

The **valid** parameter is the time in seconds that the IPv6 address exists on the interface. After the time specified expires, the IPv6 address is deleted. The default is 2592000 seconds (30 days). The **preferred** parameter must also be specified, and the valid time must be the same as or greater than the preferred time. The **valid** parameter can be specified if an IPv6 address is specified by using the **ipaddress** parameter.

**Examples** To set the interface address 3ffe::1 to be the preferred address for 300 seconds, and have a metric of 6 use the command:

```
set ipv6 int=vlan1 ip=3ffe::1 pref=300 metric=6
```

**Related Commands**

- [add ipv6 interface](#)
- [create ipv6 interface](#)
- [delete ipv6 interface](#)
- [destroy ipv6 interface](#)
- [show ipv6](#)
- [show ipv6 interface](#)

---

## set ipv6 mtu

---

**Syntax** SET IPV6 MTU=*mtu* INTERface=*interface*

where:

- *mtu* is an integer between 1280 and the maximum True MTU of the specified interface.
- *interface* is a valid interface.

**Description** This command sets the link maximum transmission unit (MTU) on the specified interface. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

**Examples** To give the vlan1 interface an MTU of 1450, use the command:

```
set ipv6 mtu=1450 int=vlan1
```

**Related Commands**

- [disable ipv6 mtudiscovery](#)
- [enable ipv6 mtudiscovery](#)
- [set ipv6 nd](#)
- [show ipv6 interface](#)

## set ipv6 nd

---

**Syntax** SET IPV6 ND INTERface=*interface* [DUPtrans=1..16]  
 [HOP=1..255] [LIFE=0|4..9000] [MAXAInt=4..1800]  
 [MCONF={YES|NO}] [MINAInt=3..1350] [MTU=1280..65535]  
 [OCONF={YES|NO}] [REAch=0..3600000]  
 [RETRans=0..4294967295]

where *interface* is a valid interface

**Description** This command allows the configuration of various Router Advertisement parameters on the specified IPv6 interface. The interface must already be assigned and configured. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

The **duptrans** parameter is the DupAddrDetectTransmits value. It is the number of Neighbour Solicitation messages that the switch sends while performing Duplicate Address Detection on a tentative address. The default is 1.

The **hop** parameter specifies the AdvCurHopLimit. This value limits the number of hops that packets can take. A value of 0 (zero) indicates that this value is not specified by this switch. The default is 64.

The **life** parameter is the AdvDefaultLifetime in seconds. This is the maximum time that the switch is the default. A host sends data to the router that is the current default router in its list of routers. A value of 0 (zero) indicates that the switch is not to be used as a default router. If the value is not 0, it must be equal to or greater than **maxaint** and should be 3 \* **maxaint**. The default is 1800.

The **maxaint** parameter specifies the MaxRtrAdvInterval, in seconds. This is the maximum time interval between unsolicited multicast Router Advertisements sent by the switch from this interface. The default is 600 seconds.

The **mconf** parameter is the AdvManaged flag. If **true** is specified, hosts receiving the advertisements use the administered stateful autoconfiguration protocol for address autoconfiguration, and request address information and other (non-address) information from the switch. The default is **false**.

Hosts ignore a change in the AdvManaged flag from **true** to **false** in the switch's advertisements when the host is already using stateful address autoconfiguration.

The **minaint** parameter specifies the MinRtrAdvInterval in seconds. This is the maximum time interval between unsolicited multicast Router Advertisements the switch sends from this interface. It should be at least 0.33 x **maxaint** but no greater than 0.75 x **maxaint**. The default is 198 seconds.

The **mtu** parameter is the AdvLinkMTU. This is the maximum transmission unit (MTU) for this link and is used by nodes that are running Path MTU Discovery. The default is 0, which indicates that this value is not specified by this switch.

The **oconf** parameter is the AdvOtherConfig flag. If **true** is specified, hosts receiving the advertisements use the administered stateful protocol for autoconfiguration of non-address information. The default is **false**. If **mconf** is **true**, **oconf** implicitly also becomes **true**.

Hosts ignore a change in the AdvOtherConfig flag from **true** to **false** in the switch's advertisements when the host is already using stateful address autoconfiguration to obtain non-address information.

The **reach** parameter is the AdvReachableTime, in milliseconds. When a node receives a reachability confirmation message as part of the Neighbour Unreachability Detection algorithm, it assumes that the switch is reachable for this length of time. The default is 0, which indicates that this timer is not specified by this switch.

The **retrans** parameter is the AdvRetransTimer, in milliseconds. This is the interval between repeats of each Router Advertisement message sent by the switch. The value you enter is rounded up to the nearest 100 milliseconds (for example, 301 becomes 400). The default is 0, which indicates that this timer is not specified by this switch.

**Examples** To set the switch to advertise a maximum transmission unit of 1450 over the vlan1 interface, use the command:

```
set ipv6 nd int=vlan1 mtu=1450
```

**Related Commands**

- [create ipv6 interface](#)
- [show ipv6 interface](#)
- [show ipv6 ndcache](#)
- [show ipv6 ndconfig](#)

## set ipv6 prefix

**Syntax** SET IPV6 PREFIX=*ipv6add/prefix-length* INTerface=*interface*  
 [AUTOonomous={YES|NO}] [ONlink={YES|NO}]  
 [PREferred=1..4294967295|INFinite]  
 [VALid=1..4294967295|INFinite]

where:

- *ipv6add* is a valid IPv6 address, with its prefix length indicated by slash notation (see [“IPv6 Interfaces and Addresses” on page 34-13](#)).
- *prefix-length* is the number between 1 and 128.
- *interface* is a valid interface.

**Description** This command modifies the settings for the specified prefix that the switch advertises on the specified interface. The prefix is not added to the interface's IPv6 address. The prefix can assist nodes on that subnet with configuration of their local-link IPv6 addresses. More than one prefix can be included in the router advertisements for a given interface. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

To see a list of current valid interfaces, use the **show ipv6 interface** command or the [show interface command on page 11-87 of Chapter 11, Interfaces](#).

If the **publish** parameter is set to **yes** for an interface by using the **add ipv6 interface** or the **set ipv6 interface** command, the prefix specified by the command is advertised as well as the prefix specified with this command.

The **prefix** parameter specifies the IPv6 prefix (e.g. 3ffe::/64) to advertise for this interface.

The **autonomous** parameter specifies whether the prefix is for the same autonomous system (for example, the Internet or an intranet). If it is, hosts use this prefix to configure their addresses over this autonomous system. The default is **yes**.

The **onlink** parameter specifies whether hosts use this prefix to configure their addresses on the local link. The default is **yes**.

The **preferred** parameter specifies the time in seconds for which this prefix is preferred over other prefixes. The prefix no longer appears in router advertisements after this period. The default is 604800 seconds (7 days).

The **valid** parameter specifies the time in seconds for which the prefix is valid. The prefix is deleted after this period. The default is 2592000 seconds (30 days).

**Examples** To make the prefix 3ffe::/64 the preferred prefix for the next 8 hours (28800 seconds), in router advertisements sent over the vlan1 interface, use the command:

```
set ipv6 prefix=3ffe::/64 int=vlan1 pref=28800
```

**Related Commands** [add ipv6 interface](#)  
[add ipv6 prefix](#)  
[delete ipv6 prefix](#)



[set ipv6 interface](#)  
[show ipv6 interface](#)  
[show ipv6 ndcache](#)

## set ipv6 route preference

---

**Syntax** SET IPV6 ROUTe PREFerence={DEFault|1..65535} PROTOcol=RIP

**Description** This command sets the IP route table preference for routes learned via the specified protocol. When more than one route in the route table matches the destination address in an IPv6 packet, the route with the lowest preference value is used to route the packet. If two or more candidate routes have the same preference, the route with the longest prefix is used. All existing dynamically learned routes in the routing table and new routes added later are updated with the specified preference value.

The **preference** parameter sets the preference for routes learned via the specified protocol. If **default** is specified, the preference reverts to the default for this protocol type.

The **protocol** parameter specifies which protocol's routing table is updated with the new preference value.

**Examples** To set RIP routes to a lower preference than all OSPF routes, use the command:

```
set ipv6 rou pref=160 prot=rip
```

**Related Commands** [show ipv6 route preference](#)

## show ipv6

**Syntax** SHow IPV6

**Description** This command displays general configuration information for the IPv6 module (Figure 34-15, Table 34-1).

Figure 34-15: Example output from the **show ipv6** command

```

IPV6 Module Configuration
-----

Module Status ..... Enabled
IPv6 Packet Forwarding ..... Enabled
IPv6 RIP ..... Disabled
IPv6 Echo Reply ..... Enabled
Name Server ..... 202.49.72.50
Secondary Name Server ..... Not Set
Source-Routed Packets ..... Discarded

Routing Protocols

RIP Neighbours ..... 0

Active Routes:

Static ..... 5
Interface ..... 1
Neighbour Discovery..... 0
RIP ..... 0
Other ..... 0
-----
Total Number of routes..... 6
Discarded routes ..... 0

```

Table 34-1: Parameters in output of the **show ipv6** command

Parameter	Meaning
Module Status	Whether the IPv6 module has been enabled or disabled.
IPv6 Packet Forwarding	Whether the switch is enabled or disabled to forward packets.
IPv6 RIP	Whether the switch has been configured to send and receive RIPv6 packets.
IPv6 Echo Reply	Whether the switch has been configured to reply to echo request (ping) packets.
Name Server	The IP address of the DNS server for name resolution.
Secondary Name Server	The IP address of a secondary DNS server, to use when the primary Name Server fails.
Source-Routed Packets	What the switch has been configured to do with source routed packets.
<b>Routing Protocols</b>	Information about the routing protocols in use.
RIP Neighbours	The number of nodes on the network known by the switch to be running RIPv6.
<b>Active Routes</b>	Information about the routes in the route table.

Table 34-1: Parameters in output of the **show ipv6** command (cont.)

Parameter	Meaning
Static	The number of static routes that have been added to the switch through manual configuration.
Interface	The number of routes that have been generated through an interface being added.
Neighbour Discovery	The number of routes gathered from neighbour discovery.
RIP	The number of routes gathered from RIPv6 packets.
Other	The number of routes gathered through other protocols, for example, OSPF.
Total Number of Routes	The total number of routes in the route table.
Discarded Routes	The number of routes that have been discarded because a better route was found through neighbour discovery. Static routes are not discarded.

**Examples** To show the general settings for the IPv6 module, use the command:

```
sh ipv6
```

**Related Commands**

- [show ipv6 counter](#)
- [show ipv6 filter](#)
- [show ipv6 host](#)
- [show ipv6 interface](#)
- [show ipv6 multicast](#)
- [show ipv6 ndcache](#)
- [show ipv6 rip](#)
- [show ipv6 route](#)
- [show ipv6 tunnel](#)

## show ipv6 counter

**Syntax** SHow IPV6 COUnter [= {ICmp | INterface}]

**Description** This command displays the IPv6 MIB counters ([Figure 34-16](#), [Table 34-2 on page 34-77](#)).

Figure 34-16: Example output from the **show ipv6 counter** command

```

IPv6 MIB Counters
-----
Interface Counters

Interface: vlan1
  InReceives ..... 0          OutForwDatagrams ..... 5
  InNoRoutes ..... 0          OutRequests ..... 5
  InDiscards ..... 0          OutDiscards ..... 0
  InAddrErrors ..... 0        OutFragOKs ..... 0
  InUnknownProtos ..... 0     OutFragFails ..... 0
  InTruncatedPkts ..... 0     OutFragCreates ..... 0
  InMcastPkts ..... 0         OutMcastPkts ..... 0
  ReasmReqds ..... 0          ReasmOKs ..... 0
  ReasmFails ..... 0
  InDelivers ..... 0
  InHdrErrors ..... 0
  InTooBigErrors ..... 0

ICMP counters
  inMsgs ..... 0              OutMsgs ..... 5
  InErrors ..... 0            OutErrors ..... 0
  InDestUnreachs ..... 0     OutDestUnreachs ..... 0
  InAdminProhibs ..... 0     OutAdminProhibs ..... 0
  InTimeExcds ..... 0        OutTimeExcds ..... 0
  InParmProblems ..... 0     OutParmProblems ..... 0
  InPktTooBigs ..... 0       OutPktTooBigs ..... 0
  InEchos ..... 0             OutEchos ..... 5
  InEchoReplies ..... 0      OutEchoReplies ..... 0
  InRouterSolicits ..... 0    OutRouterSolicits ..... 0
  InRouterAdvert ..... 0     OutRouterAdvert ..... 0
  InNeighborSolicits ..... 0  OutNeighborSolicits ..... 0
  InNeighborAdvert ..... 0   OutNeighborAdvert ..... 0
  InRedirects ..... 0         OutRedirects ..... 0
  InGroupMembQueries ..... 0  OutGroupMembQueries ..... 0
  InGroupMembResp ..... 0     OutGroupMembResp ..... 0
  InGroupMembReduct ..... 0   OutGroupMembReduct ..... 0

```

Table 34-2: Parameters in output of the **show ipv6 counter** command

Parameter	Meaning
<b>Interface Counters</b>	Counters for each IPv6 interface on the route.
Interface	Name of the IPv6 interface.
InReceives	Number of packets received on the interface.
InNoRoutes	Number of input packets discarded because no route could be found to transmit them to their destination.
InDiscards	Number of incoming packets that were discarded.
InAddrErrors	Number of packets received with invalid addresses.
InUnknownProtos	Number of packets received with unknown next headers.
InTruncatedPkts	Number of packets received that were truncated.
InMcastPkts	Number of multicast packets received.
ReasmReqds	Number of packets received that required reassembly.
ReasmFails	Number of packets that could not be reassembled.
InDelivers	Number of incoming packets that were successfully delivered to a higher layer protocol.
InHdrErrors	Number of packets received with invalid headers.
InTooBigErrors	Number of packets received that were discarded because they were too big.
OutForwDatagrams	Number of packets that have been forwarded.
OutRequests	Number of echo requests sent.
OutDiscards	Number of packet discarded messages sent.
OutFragOKs	Number of fragmentation success messages sent.
OutFragFails	Number of fragmentation failed messages sent.
OutFragCreates	Number of outgoing packet fragments generated as a result of fragmentation at this output interface.
OutMcastPkts	Number of multicast packets sent.
ReasmOKs	Number of IPv6 packets successfully reassembled. This counter is incremented at the interface to which these packets were addressed, which is not necessarily the input interface for some of the fragments.
<b>ICMP Counters</b>	Counters for ICMPv6 for all IPv6 interfaces on the switch.
inMsgs	Total number of ICMP messages received by the interface, which includes all those counted by InErrors. This interface is the interface to which the ICMP messages were addressed. This may not necessarily be the input interface for the messages.
InErrors	Number of ICMP messages that the interface received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc).
InDestUnreachs	Number of ICMP Destination Unreachable messages received by the interface.
InAdminProhibs	Number of ICMP Destination Unreachable/Communication Administratively Prohibited messages received by the interface.
InTimeExcds	Number of ICMP Time Exceeded messages received by the interface.

Table 34-2: Parameters in output of the **show ipv6 counter** command (cont.)

InParmProblems	Number of ICMP Parameter Problem messages received by the interface.
InPktTooBig	Number of ICMP Packet Too Big messages received by the interface.
InEchos	Number of ICMP Echo (request) messages received by the interface.
InEchoReplies	Number of ICMP Echo Reply messages received by the interface.
InRouterSolicits	Number of ICMP Router Solicit messages received by the interface.
InRouterAdvert	Number of ICMP Router Advertisement messages received by the interface.
InNeighborSolicits	Number of ICMP Neighbor Solicitation messages received by the interface.
InNeighbor Advert	Number of ICMP Neighbor Advertisement messages received by the interface.
InRedirects	Number of Redirect messages received by the interface.
InGroupMembQueries	Number of ICMPv6 Group Membership Query messages received by the interface.
InGroupMembResp	Number of ICMPv6 Group Membership Response messages received by the interface.
InGroupMembReduct	Number of ICMPv6 Group Membership Reduction messages received by the interface.
OutMsgs	Total number of ICMP messages that the interface attempted to send. This counter includes all those counted by OutErrors.
OutErrors	Number of ICMP messages that this interface did not send due to problems discovered within ICMP, such as a lack of buffers. This value does not include errors discovered outside the ICMP layer, such as the inability of IPv6 to route the resulting packet.
OutDestUnreachs	Number of ICMP Destination Unreachable messages sent by the interface.
OutAdminProhibs	Number of ICMP destination Unreachable/Communication Administratively Prohibited messages sent by the interface.
OutTimeExcds	Number of ICMP Time Exceeded messages sent by the interface.
OutParmProblems	Number of ICMP Parameter Problem messages sent by the interface.
OutPktTooBig	Number of ICMP Packet Too Big messages sent by the interface.
OutEchos	Number of ICMP Echo (request) messages sent by the interface.
OutEchoREplies	Number of ICMP Echo Reply messages sent by the interface.
OutRouterSolicits	Number of ICMP Router Solicit messages sent by the interface.
OutRouterAdvert	Number of ICMP Router Advertisement messages sent by the interface.

Table 34-2: Parameters in output of the **show ipv6 counter** command (cont.)

OutNeighborSolicits	Number of ICMP Neighbor Solicit messages sent by the interface.
OutNeighborAdvert	Number of ICMP Router Advertisement messages sent by the interface.
OutRedirects	Number of Redirect messages sent by the interface. For a host, this object is always zero since hosts do not send redirects.
OutGroupMembQueries	Number of ICMPv6 Group Membership Query messages sent by the interface.
OutGroupMembResp	Number of ICMPv6 Group Membership Response messages sent by the interface.
OutGroupMembReduct	Number of ICMPv6 Group Membership Reduction messages sent by the interface.

**Examples** To show the IPv6 counters, use the command:

```
sh ipv6 cou
```

**Related Commands**

- [show ipv6](#)
- [show ipv6 filter](#)
- [show ipv6 host](#)
- [show ipv6 interface](#)
- [show ipv6 multicast](#)
- [show ipv6 ndcache](#)
- [show ipv6 rip](#)
- [show ipv6 route](#)
- [show ipv6 tunnel](#)

## show ipv6 filter

**Syntax** SHow IPV6 FILter[=*filter-id*]

where *filter-id* is a number from 0 to 99 or 200 to 299

**Description** This command displays information about filters. If a filter is specified, the entries in the filter are displayed. If a filter is not specified, the entries in all filters are displayed ([Figure 34-17](#), [Table 34-3 on page 34-81](#)).

Figure 34-17: Example output from the **show ipv6 filter** command.

IP Filters									
No.	Ent.	Source Address	/Plen						
		Source Port							
		Destination Address	/Plen						
		Destination Port							
		Size			Prot. (C/T)				
		Options					Session		
		Logging							
		Matches					Act/Pri		
-----									
1	1	FE80:0000:0000:0000:0260:97FF:FE8F:64AA	/16						
		Any							
		FE80:0000:0000:0000:0324:96BB:FE8F:64AA	/32						
		Any							
		Any					Any		
		No					Any		
		None							
		0					Exclude		
-----									
	2	FE80:0000:0000:0000:0260:97FF:FE8F:63CC	/16						
		Any							
		::					/16		
		Any							
		Any			ICMP Any		Any		
		No					Any		
		Header							
		155					Include		
		Passes: 0	Fails: 636						
-----									
2	1	FE80:0000:0000:0000:0333:97FF:FE8F:64AA	/32						
		Any							
		FE80:0000:0000:0000:0444:96BB:FE8F:64AA	/32						
		Any							
		Any			ICMP Any		Any		
		No					Any		
		Header							
		132					Exclude		
		Passes: 0	Fails: 0						



Table 34-3: Parameters in output of the **show ipv6 filter** command

Parameter	Meaning
Source Address	Source IPv6 address.
Source Port	Source IPv6 port.
Plen	Number of leftmost contiguous bits of the address to match against.
Destination Address	Destination IPv6 address.
Destination Port	Destination IPv6 port.
Size	Size of the packet to match against.
Prot. (C/T)	Protocol to match against. If the protocol is ICMP, the ICMP type and code are shown.
Options	Whether the Hop by Hop or Destination Option extension headers are present.
Session	Status of the TCP session establishment process.
Logging	Log details of the packet.
Matches	Number of packets that matched the entry.
Act/Pri	Action to take or priority to give if packets match the entry.

**Examples** To show all filters, use the command:

```
sh ipv6 fil
```

To show all entries in filter 2, use the command:

```
sh ipv6 fil=2
```

**Related Commands** [add ipv6 filter](#)  
[delete ipv6 filter](#)  
[set ipv6 filter](#)

## show ipv6 host

**Syntax** SHow IPV6 HOsT

**Description** This command uses the **host** parameter to display information on the host names associated with IPv6 addresses on the switch ([Figure 34-18](#), [Table 34-4](#)).

Figure 34-18: Example output from the **show ipv6 host** command.

IPv6 Address	Host Name
3ffe::0002	foobar
3ffe::0004	mainserver
3ffe::0006	bob

Table 34-4: Parameters in output of the **show ipv6 host** command

Parameter	Meaning
IPv6 Address	IPv6 address of the host.
Host Name	Alias assigned to that host.

**Examples** To show the host names associated with the IPv6 addresses on the switch, use the command:

```
sh ipv6 ho
```

**Related Commands** [add ipv6 host](#)  
[delete ipv6 host](#)

## show ipv6 interface

**Syntax** SHow IPV6 INTeRface[=*interface*]

**Description** This command displays information about the interfaces configured for IPv6. If an interface is specified then information for the specified interface is displayed, otherwise information for all IPv6 interfaces is displayed (Figure 34-19, Table 34-5 on page 34-84).

Figure 34-19: Example output from the **show ipv6 interface** command

```

IPv6 Interface Configuration
-----
Interface ..... ppp0
Ipv6 Interface Index ..... 1
Link-layer address ..... 00-00-00-00-00-00
EUI-64 Interface Identifier ..... 0200CDFFFE01F9C1
IPSec ..... No
True MTU/Link MTU ..... 1280/1280
Multicast status ..... Enabled
Send Router Advertisements ? ..... No
Ipv6 Interface Addresses :
  Int  Addresses
  Type  Scope  State      Enabled  Valid      PLen  Decrement
                                     Preferred  Publish
-----
  0      fe80::0200:cdff:fe01:f9c1
    unicast link preferred Yes      infinite infinite No
  1      2004::0001
    unicast global preferred Yes      infinite infinite Yes

IPv6 Interface Configuration
-----
Interface ..... vlan1
Ipv6 Interface Index ..... 2
Link-layer address ..... 00-00-cd-01-f9-c1
EUI-64 Interface Identifier ..... 0200CDFFFE01F9C1
IPSec ..... No
True MTU/Link MTU ..... 1500/1500
Multicast status ..... Enabled
Send Router Advertisements ? ..... No
Ipv6 Interface Addresses :
  Int  Addresses
  Type  Scope  State      Enabled  Valid      PLen  Decrement
                                     Preferred  Publish
-----
  0      fe80::0200:cdff:fe01:f9c1
    unicast link preferred Yes      infinite infinite No
  1      2001::0001
    unicast global preferred Yes      infinite infinite Yes
-----

```

Table 34-5: Parameters in output of the **show ipv6 interface** command

Parameter	Meaning
Interface	The name of the physical or virtual interface.
IPv6 Interface Index	The index of the interface in the IPv6 interface table.
Link-layer address	The link layer address of the interface.
EUI-64 Interface Identifier	The interface identifier in IEEE EUI-64 format.
IPSec	Whether any IPSec policies are attached to the interface.
True MTU/Link MTU	The Maximum Transmission Unit of the interface. The "True MTU" is the true link MTU of the interface. The "Link MTU" is the MTU set by the user.
Multicast Status	Whether multicast packet reception is enabled on the interface.
Send Router Advertisements?	Whether the interface sends router advertisements.
<b>IPv6 Interface Addresses</b>	
Int	The index of this address in the IPv6 address table.
Addresses	The IPv6 addresses configured on the switch.
Plen	The prefix length of the address in bits.
Decrement	Whether address lifetimes decrement.
Type	Whether the address is unicast or anycast.
Scope	The scope of the address; either Link, Site, or Global.
State	The status of the address according to duplicate address detection; either "preferred" if the address is unique or "deprecated" if the address is a duplicate or has timed out.
Enabled	Whether the address is enabled.
Valid	The time in seconds for which this address is valid.
Preferred	The time in seconds for which this address is the interface's preferred address.
Publish	Whether the switch includes this prefix in Router Advertisement messages.

**Examples** To display information about the vlan1 interface, use the command:

```
sh ipv6 int=vlan1
```

**Related Commands**

- [add ipv6 interface](#)
- [delete ipv6 interface](#)
- [set ipv6 interface](#)
- [show ipv6](#)

## show ipv6 multicast

**Syntax** SHow IPV6 MULticast

**Description** This command displays information about the multicast memberships on the switch ([Figure 34-20](#), [Table 34-6](#)).

Figure 34-20: Example output from the **show ipv6 multicast** command

```
Ipv6 Multicast Memberships:
Multicast Address          Interface
-----
ff02::0001:ff01:f9f5      vlan1
ff02::0002                vlan1
ff02::0001                vlan1
ff02::0001:ff00:0006      vlan1
-----
```

Table 34-6: Parameters in output of the **show ipv6 multicast** command

Parameter	Meaning
Multicast Address	The address of a multicast group.
Interface	The interface that belongs to the multicast group.

**Examples** To show the multicast settings, use the command:

```
sh ipv6 mul
```

**Related Commands** [show ipv6](#)

## show ipv6 ndcache

**Syntax** SHow IPV6 NDCaChe

**Description** This command displays information gathered through neighbour discovery (Figure 34-21, Table 34-7).

Figure 34-21: Example output from **show ipv6 ndcache** command

```
Ipv6 Neighbour Cache:
Ipv6 Address                               Link-layer address
Interface      State      LastReachble IsRouter
-----
3ffe:0c00:8017:0120:0220:35ff:feb1:7065  00-20-35-b1-70-65
vlan1          stale      0 msec      no
fe80::0220:35ff:feb1:7065                00-20-35-b1-70-65
vlan1          stale      0 msec      no
-----
```

Table 34-7: Parameters in output of the **show ipv6 ndcache** command

Parameter	Meaning
IPv6 Address	IPv6 address of the neighbour.
Link-layer address	Link-layer address of the neighbour.
Interface	The name of the interface through which the neighbour is reachable.
State	Whether the neighbour discovery is reachable, unreachable, or stale.
LastReachable	Number of milliseconds within which the neighbour was last reachable.
IsRouter	Whether the neighbour is a router.

**Examples** To show neighbour information, use the command:

```
sh ipv6 ndca
```

**Related Commands** [disable ipv6](#)  
[enable ipv6](#)

## show ipv6 ndconfig

**Syntax** SHow IPV6 NDConfig [INTErface=*interface*]

where *interface* is a valid interface

**Description** This command displays various Neighbour Discovery parameters (Figure 34-22, Table 34-8). If an IPv6 interface is specified, only parameters on that interface are displayed. Valid interfaces are:

- PPP (such as ppp0)
- VLAN (such as vlan1)
- virtual tunnel (such as virt9)

Figure 34-22: Example output from the **show ipv6 ndconfig** command

```

Ipv6 Neighbour Discovery Information:
-----
Interface : vlan1
AdvSendAdvertisements ..... No
MaxRtrAdvInterval .....600 seconds
MinRtrAdvInterval ..... 198 seconds
AdvManagedFlag ..... False
AdvOtherConfigFlag ..... False
AdvLinkMTU ..... 1500
AdvReachableTime ..... 0 milliseconds
AdvRetransTimer ..... 0 milliseconds
AdvCurHopLimit ..... 64
AdvDefaultLifetime ..... 54 seconds
AdvPrefixList plen valid/pref A/O
-----
3ffe:0002:: /64 infinite/infinite 1/1
3ffe:0501:ffff:0100:: /64 2591904/604704 1/1
-----

```

Table 34-8: Parameters in output of the **show ipv6 ndconfig** command

Parameter	Meaning
Interface	Name of the IPv6 interface to the neighbour.
AdvSendAdvertisements	Whether the switch sends Router Advertisements.
MaxRtrAdvInterval	Maximum number of seconds between unsolicited multicast Router Advertisements sent by the switch.
MinRtrAdvInterval	Minimum number of seconds between unsolicited multicast Router Advertisements sent by the switch.
AdvManagedFlag	Whether hosts receiving the advertisements use the administered stateful autoconfiguration protocol for address autoconfiguration.
AdvOtherConfigFlag	Whether hosts receiving the advertisements use the administered stateful autoconfiguration protocol for autoconfiguration of non-address information.
AdvLinkMTU	Maximum transmission unit (MTU) for this link.
AdvReachableTime	Time in seconds after receiving the Router Advertisement that a node assumes the switch is reachable.
AdvRetransTimer	The interval in milliseconds between repeats of each Router Advertisement message sent by the switch.

Table 34-8: Parameters in output of the **show ipv6 ndconfig** command (cont.)

Parameter	Meaning
AdvCurHopLimit	Current Hop Limit, which is used to limit the number of hops that packets can take.
AdvDefaultLifetime	Maximum time in seconds that the switch is the default router.
AdvPrefixList	Prefixes that are included in the Router Advertisements.
plen	The prefix length in bits.
valid	How long in seconds that the prefix is valid. The prefix is deleted after this period.
pref	How long in seconds that the preferred prefix is valid. The prefix is no longer advertised after this period.
A	Whether the prefix is for the same autonomous system as the neighbour (for example, the Internet or an intranet).
O	Whether the prefix is on the same link as the neighbour.

**Example** To display neighbour discovery parameters for the vlan1 interface

```
sh ipv6 ndco int=vlan1
```

**Related Commands**

- [add ipv6 prefix](#)
- [set ipv6 nd](#)
- [set ipv6 prefix](#)



## show ipv6 rip

**Syntax** SHow IPV6 RIP [{COUnTer|TImEr}]

**Description** This command displays information about RIP interfaces:

- If neither **counter** or **timer** is specified, summary information is displayed about the interfaces on which RIP is enabled (Figure 34-23, Table 34-9).
- If **timer** is specified, information about the timers for each route is displayed (Figure 34-24, Table 34-10).
- If **counter** is specified, counters for RIP are displayed (Figure 34-25 on page 34-90, Table 34-11 on page 34-90).

Figure 34-23: Example output from the **show ipv6 rip** command

Interface	Poison	Ipv6 Address
virt0	Yes	3ffe::0002

Table 34-9: Parameters in output of the **show ipv6 rip** command

Parameter	Meaning
Interface	Name of the IPv6 interface over which RIP is running.
Poison	Whether poison reverse is enabled on the interface.
IPv6 Address	IPv6 address of the interface.

Figure 34-24: Example output from the **show ipv6 rip timer** command

RIPng route timers:					
Destination	int.	met.	val	hold	flush
2fff::/32	1	2	18	0	28
2ffa::/32	1	2	18	0	28
2ffb::/32	1	2	18	0	28

Table 34-10: Parameters in output of the **show ipv6 rip timer** command

Parameter	Meaning
Destination	Destination network for the route.
int	Name of the interface over which RIP is running.
met	RIP metric associated with this route.
val	Valid lifetime of the route, in seconds.
hold	Interval in seconds after the route becomes invalid, during which the switch ignores updates for the route that would normally make the route valid again.
flush	Interval in seconds from the last update of the route, until the route is flushed from the route table.

Figure 34-25: Example output from the **show ipv6 rip counter** command

```

IPV6 RIPng Counter Summary:
  Input:
    inResponses ..... 3
    inDiscards ..... 0
  Output:
    outResponses..... 0

```

Table 34-11: Parameters in output of the **show ipv6 rip counter** command

Parameter	Meaning
inResponses	Number of RIP packets received.
inDiscards	Number of RIP packets that were received and discarded.
outResponses	Number of RIP packets sent.

**Examples** To display a list of RIP interfaces, use the command:

```
sh ipv6 rip
```

To display a list of RIP timers, use the command:

```
sh ipv6 rip tim
```

To display a list of RIP counters, use the command:

```
sh ipv6 rip cou
```

**Related Commands**

- [add ipv6 rip](#)
- [delete ipv6 rip](#)
- [disable ipv6 rip](#)
- [enable ipv6 rip](#)
- [show ipv6](#)

## show ipv6 route

**Syntax** SHow IPV6 ROUTe [FULL]

**Description** This command displays the contents of the IPv6 route table (Figure 34-26, Table 34-12). The **full** parameter displays all the routes in the IPv6 route table. When the **full** parameter is not specified, then the command displays a subset of the routing table that includes:

- all static routes
- all interface routes
- only the RIP routes that are alive and best

Figure 34-26: Example output from the **show ipv6 route** command

Destination prefix	Int.	Age	Policy	Protocol	Next Hop Metric	Pref	Tunnel	Flags
2ffe::/32	virt0	no	0	static	2	60	yes	-
2fff::/32	virt0	yes	0	ripng	2	100	yes	-
2ffa::/32	virt0	yes	0	ripng	2	100	yes	-
2ffb::/32	virt0	yes	0	ripng	2	100	yes	-

Codes: P=publish, Do=down

Table 34-12: Parameters in output of the **show ipv6 route** command

Parameter	Meaning
Destination prefix	The destination network for the route.
Next Hop	The next node along the path to the destination.
Int.	The interface via which packets are sent.
Age	Whether the route ages out.
Policy	Reserved for future use.
Protocol	The protocol where the route was gathered: <ul style="list-style-type: none"> <li>static - the route was configured manually via the command line</li> <li>interface - the route was automatically created through adding an interface</li> <li>ripng - the route was determined via RIPv6</li> </ul>
Metric	The RIP metric that is being used. This usually indicates how far away the destination is, in hops.
Pref	The preference of the route over others to the same destination.
Tunnel	Whether the interface is a tunnel; one of “yes” or “no”.
Flags	The various flags that have been set or implied on the route.

**Examples** To show all the switch’s IPv6 routes, use the command:

```
sh ipv6 rou full
```

**Related Commands**   [add ipv6 route](#)  
[delete ipv6 route](#)

## show ipv6 route multicast

**Syntax** SHow IPV6 ROUte MULticast [=*ipv6address*]  
[SOurce=*ipv6address*]

where *ipv6address* is an IPv6 address, with or without the wildcard character (\*). Examples of valid addresses are given below. The wildcard cannot be used in addresses in other IPv6 commands.

**Description** This command displays information about the IPv6 multicast forwarding table. If no optional parameters are specified, the contents of the forwarding table are displayed in full ([Figure 34-27](#), [Table 34-13 on page 34-94](#)).

If a multicast group address is specified in the **multicast** parameter, all packet sources that send to the multicast group address are displayed, as well as upstream and downstream interfaces.

If the **source** parameter is specified, then multicast groups sourced from the specified IP address are displayed.

In both the **multicast** and **source** parameters, the wildcard character "\*" may replace any complete octet in the address, but not part of an octet. The following examples are valid uses of the wildcard:

```
1111:1111:1111:1111:22*:1111:1111:1111
```

```
1111:1111:1111:1111:*33:1111:1111:1111
```

If both octets in a pair are to be wildcards, a single wildcard character can be used, so the following two addresses are equivalent:

```
1111:1111:1111:1111:**:1111:1111:1111
```

```
1111:1111:1111:1111*:1111:1111:1111
```

An address can contain as many wildcards as required, but must conform to the IPv6 address syntax convention of 8 pairs of octets separated by colons:

```
1111:*:*:*22:1111:1111:11*:1111
```

```
*:*:*:1111::
```

Figure 34-27: Example output from the **show ipv6 route multicast** command

```
Multicast Address ..... ff05:1111:1111:1111:1111:1111:1111:1111
Source/Prefix ..... 1111::1111/128
Scope ..... Site
Protocol ..... PIM
InPort ..... ppp0
OutPorts ..... eth0, eth1
Source/Prefix ..... ff0e:1111:1111:1111:1111:1111:1111:1111/128
Scope ..... Global
Protocol ..... PIM
InPort ..... ppp0
OutPorts ..... eth1
Multicast Address ..... ff05:1111:1111:1111:1111:1111:1111:2222
Source/Prefix ..... 1111:1111:1111:1111:1111:1111:1111:1111/128
Scope ..... Site
Protocol ..... PIM
InPort ..... eth0
OutPorts ..... ppp2
```

Table 34-13: Parameters in output of the **show ipv6 route multicast** command

Parameter	Meaning
Source/Prefix	IPv6 address of the host that is sourcing multicast datagrams addressed to the specified groups.
Multicast Address	IPv6 address to which multicast datagrams are addressed.
Scope	Whether the multicast address is set for site or global.
Protocol	Multicast routing protocol that contributes this forwarding entry.
InPort	Incoming (upstream) port for the (source, group) pair.
OutPorts	Outgoing (downstream) ports over which multicast datagrams for the (source, group) pair are forwarded.

**Examples** To display the complete IPv6 multicast forwarding table, use the command:

```
sh ipv6 rou mul
```

To display the IPv6 multicast forwarding table for packet sources that send to all groups that begin with ff05 and have 2a as the 10th octet, use the command:

```
sh ipv6 rou mul=ff05:*:*:2a:*:*:*
```

**Related Commands** [show ipv6](#)

## show ipv6 route preference

**Syntax** SHow IPV6 ROUTe PREference

**Description** This command displays information about the current IP route table preferences for the routing protocol ([Figure 34-28](#), [Table 34-14](#)).

Figure 34-28: Example output from the **show ipv6 route preference** command

IPv6 Route Preference	
-----	
Protocol	Preference
-----	
RIP .....	100 (default)
-----	

Table 34-14: Parameters in output of the **show ipv6 route preference** command

Parameter	Meaning
Protocol	Available routing protocols
Preference	Preference value for the routing protocol - a larger preference value indicates a less desirable routing protocol.

**Related Commands** [set ipv6 route preference](#)

# show ipv6 tunnel

**Syntax** SHow IPV6 TUNnel

**Description** This command displays information about the tunnels that are configured on the switch ([Figure 34-29](#), [Table 34-15](#)).

Figure 34-29: Example output from the **show ipv6 tunnel** command

```
Ipv6 Tunnels:

Interface   Ipv6 Tunnel Address
  Tunnel start          Tunnel end
-----
virt0       fe80::c0a8:0102
  192.168.1.2          192.168.1.1
-----
```

Table 34-15: Parameters in the **show ipv6 tunnel** command

Parameter	Meaning
Interface	Interface of the tunnel. Tunnels always have an interface name of <i>virt</i> . Unless an interface number was specified when the tunnel was created, they are numbered from 0 in order of creation.
IPv6 Tunnel Address	IPv6 address that the switch assigned to the tunnel interface when the tunnel was created.
Tunnel Start	IPv4 address of the start of the tunnel.
Tunnel End	IPv4 address of the end of the tunnel.

**Examples** To show the tunnels configured on the switch, use the command:

```
sh ipv6 tun
```

**Related Commands** [add ipv6 tunnel](#)  
[delete ipv6 tunnel](#)

## show ipv6 udp

**Syntax** SHow IPV6 UDP

**Description** This command displays the state of current UDP sessions over IPv6 ([Figure 34-30](#), [Table 34-16](#)).

Figure 34-30: Example output of the new **show ipv6 udp** command

Local port	Local address	Remote port	Process
51650	fe81::230:84ff:fe6a:ef68	6219	TFTP

Table 34-16: Parameters in the output of the **show ipv6 udp** command

Parameter	Meaning										
Local Port	The UDP port number used for the UDP session on this switch.										
Local Address	The IPv6 address of the last interface that was used to transport UDP packets from the switch for the given process. A blank address indicates that the UDP session is active, but either no packets have been transmitted yet, or packets have been transmitted without specifying the source IP address.										
Remote Port	The UDP port number used for the UDP session on the remote device. A value of zero indicates that UDP packets from any remote port will be accepted for the session.										
Process	The process that is using the UDP session. The following process types may use UDP on the switch: <table> <tr> <td>TFTP</td><td>Download/upload of files using the Trivial File Transfer Protocol</td></tr> <tr> <td>DHCP SVR</td><td>External network node configuration by the switch acting as a Dynamic Host Configuration Protocol Server</td></tr> <tr> <td>DHCP CLT</td><td>Communications by the switch when acting as a client, using the Dynamic Host Configuration Protocol</td></tr> <tr> <td>RIP</td><td>Routing of IP packets using the Routing Information Protocol</td></tr> <tr> <td>ISAKMP</td><td>Secure communications using the Internet Security Association and Key Management Protocol</td></tr> </table>	TFTP	Download/upload of files using the Trivial File Transfer Protocol	DHCP SVR	External network node configuration by the switch acting as a Dynamic Host Configuration Protocol Server	DHCP CLT	Communications by the switch when acting as a client, using the Dynamic Host Configuration Protocol	RIP	Routing of IP packets using the Routing Information Protocol	ISAKMP	Secure communications using the Internet Security Association and Key Management Protocol
TFTP	Download/upload of files using the Trivial File Transfer Protocol										
DHCP SVR	External network node configuration by the switch acting as a Dynamic Host Configuration Protocol Server										
DHCP CLT	Communications by the switch when acting as a client, using the Dynamic Host Configuration Protocol										
RIP	Routing of IP packets using the Routing Information Protocol										
ISAKMP	Secure communications using the Internet Security Association and Key Management Protocol										

**Examples** To see whether any UDP sessions are active over IPv6 and which ports they are using, use the command:

```
show ipv6 udp
```

**Related Commands** [show ipv6](#)