

# IPv6 Technical Reference

## (Microsoft Corporation)

The current Internet has fueled significant innovation and growth in network computing. To support this growth and the demand for more collaborative communication experiences, the current version of the Internet Protocol (IP), called IP Version 4 (IPv4), is being replaced with a new standard.

IPv4 is an agreed-upon set of protocols, or rules, that allow computers to communicate with each other by specifying the format of packets and the addressing scheme.

The new version of IP, called IP version 6 (IPv6), performs the same functions as IPv4 but resolves unanticipated IPv4 design issues.

As a core networking protocol in the Microsoft Windows Server 2003 operating systems, IPv6 serves as one of two Internet protocols that enable computers that are running Microsoft Windows operating systems to communicate on intranets and over the Internet.

This subject describes how IPv6 relates to other networking protocols, the functions that IPv6 performs, how IPv6 addresses are structured and assigned, how IPv6 packets are structured and routed, and how IPv6 can interoperate with IPv4.

### Introduction to IPv6

Microsoft is delivering support for the emerging update to the Internet Protocol, commonly referred to as IP version 6 — or simply IPv6 (RFC 2460). This protocol suite is based on a standard from the Internet Engineering Task Force (IETF), and it is designed to significantly increase the size of the address space used to identify communication endpoints in the Internet, thereby allowing it to continue its tremendous growth rate. As a core networking protocol in Windows Server 2003, IPv6 serves as one of two Internet protocols that enable computers running Windows to communicate on intranets and over the Internet.

The recent broad adoption of always-on technologies such as Digital Subscriber Line (DSL) and cable modems, coupled with the pending integration of personal data assistants (PDAs) and cellular phones into always-addressable Mobile Information Appliances, significantly elevates the urgency to expand the address space that Internet-connected systems use to communicate. The address space currently used is defined as part of the Internet Protocol, or IP (the network layer of the TCP/IP protocol suite).

The version of IP commonly used today, Version 4 (IPv4), has not been substantially changed since RFC 791 was published in 1981. Since then, IPv4 has proven to be robust, easily implemented, and interoperable. It has withstood the test of scaling an internetwork (a network of networks) to a global utility the size of today's Internet.

IPv6 will continue the tradition of the IPv4 protocol, which gained much of its acceptance by defining mechanisms to tie systems together over a wide variety of disparate networking technologies. Already defined link-layer mappings for transporting IPv6 include Ethernet, Point-to-Point Protocol (PPP), Fiber Distributed Data Interface (FDDI), Token Ring, Asynchronous Transfer Mode (ATM), Frame Relay, and IEEE 1394. From an architectural perspective, an IPv4-based infrastructure appears to systems that are enabled for IPv6 as a single segment non-broadcast multi-access (NBMA) network. The capability to send IPv6 traffic over existing IPv4 networks will provide an initial reach as broad as the current Internet, limited only by the endpoints' ability and readiness to make use of it.

New capabilities that are expected to drive rapid adoption include scoped addresses; stateless autoconfiguration, which lowers complexity and management burden; and mandatory IP security (IPSec), which permits end-to-end data authentication and integrity and increases privacy of connections. In addition, technologies that extend the lifetime of IPv4 (such as network address

## **IPv6 Technical Reference** (Microsoft Corporation)

translation, or NATs) frequently do not work with existing applications, and those technologies are already restricting the flexibility to deploy new applications. NATs are popular today because they allow multiple systems to share a single public IPv4 address. However, they tend to enforce a client/server usage model where the client uses private address space with only the server existing in public address space. IPv6 brings back the capability of “end-to-end control of communications,” making networking applications simpler as the network again becomes transparent.

### **IPv6 in Windows Server 2003**

Wireless technologies are emerging in ways that make ad-hoc networks between personal devices more feasible. Setting up systems to work in an ad-hoc mode is challenging enough, but many of these personal devices will also need to perform in the managed environment of the workplace. Switching between these modes is frequently frustrating and is significantly more involved than using either mode on its own. To reduce the complexity, IPv6 has defined an architectural principle that systems are required to simultaneously support multiple addresses. Coupling this capability with scoped addresses results in the ability to move easily and automatically between ad-hoc and managed environments. The IPv6 implementation will automatically adapt itself to the current needs, be it ad-hoc, home, or business connections.

To address concerns about security and privacy, the Microsoft IPv6 implementation includes IPSec, which provides data authenticity, data integrity, and data confidentiality across the array of protocols used by the various applications. Providing the capability at the network layer frees developers from having to add specific security capabilities to every application.

In addition, Microsoft helped standardize temporary addresses. To make stateless autoconfiguration work well and to ensure global uniqueness, the standards community chose the underlying hardware address (the MAC address) for use as part of an IPv6 address. The side effect of this approach is that all communications are traceable to the specific hardware device.

Although it is technically necessary to have a published (over some scope), globally unique address to receive incoming connections, the address of an originator requires only current global uniqueness (not publication). To alleviate this potential privacy concern, Microsoft has authored RFC 3041 to define a locally generated address mechanism where the result is valid only for a period that the local system or application determines. The ability of IPv6 systems to simultaneously support multiple addresses allows each application to use an independent address, an application to use a different address for each service to which it connects, or both.

Peer-to-peer applications that are made easier using IPv6 include IP telephony and video teleconferencing. These and similar applications are likely to take advantage of the Quality of Service (QoS) features defined for IPv6. Although many QoS features have also been defined as add-ons for IPv4, the mechanism selected was to redefine the meaning for the Type of Service field of the IP header, which caused collisions with historical implementations. The effort to provide QoS for IPv4 has been a struggle due to differing models of deployment. This effort is not wasted though, because it is forcing many details to be worked through — from hardware capabilities to business practices. Systems that are enabled for IPv6 will be able to leverage this effort to provide an array of service levels that are consistent from end to end.

### **Transitioning from IPv4 to IPv6**

The conversion from IPv4 to IPv6 will be a larger task for the industry than was the preparation for year 2000. This protocol change will affect nearly all networked applications, end systems, infrastructure systems, and network architectures. This change must be approached with responsibility to prevent the costly, unproductive missteps that often result from broad, premature

## IPv6 Technical Reference (Microsoft Corporation)

availability of technologies. Unlike the year 2000 issue, the conversion to IPv6 has no specific timeline. However, as noted earlier, the rate of IPv4 address consumption is rapidly increasing. Simplicity of deployment will be the key to rapid adoption.

Like IPv4 (where early deployments frequently transited X.25 networks), IPv6 deployment will start at the edge of the network, taking advantage of framing within any available network technology. Internet service providers (ISPs) will deploy native IPv6 routing based on customer demand. However, this conversion may be slow because ISPs will need several years to replace network equipment. Microsoft is taking the approach that encapsulating IPv6 packets within IPv4 will allow incremental deployments of end systems that will, in turn, demonstrate the demand to the ISPs.

To stay on the high performance path of the existing routers, computers that are running Windows and that are enabled for IPv6 will default to tunneling over IPv4 unless the ISP provides a specific indication to do otherwise and a native IPv6 path exists end to end. The only requirement is that systems that are directly connected to an ISP must receive at least one public IPv4 address. (The address ranges specified in RFC 1918 are not public.) Other systems in a home or business will receive 6to4 (RFC 3056) prefix Router Advertisement messages from the directly connected system. In the presence of NATs that are not enabled for IPv6 where only private addresses are available, a supplementary technology will be used. This technology will tunnel IPv6 traffic over NATs by including a User Datagram Protocol (UDP) header that can be used to provide a mechanism for 6to4-type tunneling across the IPv4 Internet. In enterprise environments, an incremental upgrade to IPv6 is possible using the Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). ISATAP allows IPv6-only hosts and subnets to fully co-exist and interoperate with IPv4 hosts and subnets in an intranet. In partnership with 6to4 technology, a comprehensive incremental migration solution is available to businesses that are taking their corporate networks through this transition.

Despite these approaches, the transition will not be easy. Most manufacturers will produce systems that support both IPv4 and IPv6 so that, if connections are not possible using IPv6, the systems can fall back and succeed using IPv4 (if IPv4 connectivity existed before the introduction of IPv6). The overall goal is to ensure a smooth transition and deployments where updated applications can take advantage of the new protocol without breaking existing functionality. To this end, new Windows APIs have been defined to specifically isolate the legacy applications from unintentional exposure to protocol differences, including the larger IPv6 addresses.

Microsoft has taken four key steps to deliver IPv6:

- In 1998, Microsoft Research released an IPv6 implementation to help the community that was developing standards for IPv6 understand and test the protocol during its definition.
- In March 2000, a technology preview was released for computers that were running Windows 2000. This release helped developers become familiar with the concepts and capabilities they would encounter when they enabled their applications to use IPv6.
- In October 2001, Windows XP was released with a developer preview IPv6 stack and key components of the system enabled for IPv6 so developers could begin the task of enabling their applications for IPv6. It is also expected that early adoption customers will start using IPv6 in test labs. This testing will allow those customers to have better visibility into managing their eventual rollout, and it will help identify any issues that need to be addressed in networked products.
- In November 2001, beta 3 for Windows Server 2003 was released with the first edition of the Microsoft production stack and components enabled for IPv6.

# IPv6 Technical Reference

## (Microsoft Corporation)

Current releases, including both Windows Server 2003 and Windows XP with Service Pack 1, provide general availability of this stack. Customers can begin production rollout of systems and applications that take advantage of this protocol.

The Microsoft implementation of IPv6 is easy to deploy because it includes stateless address autoconfiguration (including temporary addresses), automatic tunneling over existing IPv4 networks, and appropriate use of scoped addresses.

Because IP is a fundamental and pervasive technology within the operating system, it is not feasible to retrofit versions of Windows prior to Windows Server 2003 and Windows XP. However, to maintain backwards compatibility, versions of Windows that are enabled for IPv6 will also provide the capability to natively communicate using IPv4 for the foreseeable future. Although translation between IPv4 and IPv6 will be necessary in some cases (such as late in a transition when new IPv6-only devices need to access yet-to-be-retired IPv4-only systems), it is not expected to be the norm for early deployments. Whenever the IPv6-only devices arrive, the issues that surround address translation are typically specific to a given application. Thus as they arise, these scenarios will require targeted development on a case-by-case basis.

### Limitations of IPv4

IPv4's initial design did not anticipate the following:

- The recent exponential growth of the Internet and the impending exhaustion of the IPv4 address space. Because IPv4 addresses have become relatively scarce, some organizations have been forced to use NATs to map multiple private addresses to a single public IP address. NATs promote reuse of the private address space, but they do not support standards-based network-layer security or the correct mapping of all upper layer protocols. NATs can also create problems when they connect two organizations that use the private address space. Additionally, the rising prominence of Internet-connected devices and appliances ensures that the public IPv4 address space will eventually be depleted.
- The requirement for security at the Internet layer. Private communication over a public medium such as the Internet requires encryption services that protect the data being sent from being viewed or modified in transit. IPsec provides security for IPv4 packets, but this standard is optional, and proprietary solutions prevail.
- The growth of the Internet and the ability of Internet backbone routers to maintain large routing tables. Because of the way that IPv4 network IDs have been and are currently allocated, the routing tables of Internet backbone routers routinely contain more than 85,000 routes. The routing infrastructure of the IPv4 Internet combines both flat and hierarchical routing.
- The need to better support real-time delivery of data — also called quality of service (QoS). QoS standards exist for IPv4, but real-time traffic support relies on the IPv4 Type of Service (TOS) field and the identification of the payload, which is typically done using a UDP or TCP port. Unfortunately, the IPv4 TOS field has limited functionality, and various local interpretations developed over time. In addition, payload identification using a TCP and UDP port is not possible when the payload is encrypted.

### IPv6 Features that Fix IPv4 Limitations

IPv6 includes the following features:

# IPv6 Technical Reference

## (Microsoft Corporation)

- New header format
- Larger address space
- Efficient and hierarchical addressing and routing infrastructure
- Stateless and stateful address configuration
- Built-in security
- Better support for QoS
- New protocol for neighboring node interaction
- Extensibility

The following subsections discuss each of these new features in detail.

### **New Header Format**

The IPv6 header has a new format that is designed to minimize header overhead. This optimization is achieved by moving both non-essential fields and optional fields to extension headers that appear after the IPv6 header. Intermediate routes can process the streamlined IPv6 header more efficiently. IPv4 headers and IPv6 headers do not interoperate. IPv6 is not a superset of functionality that is backward compatible with IPv4. A host or router must use an implementation of both IPv4 and IPv6 to recognize and process both header formats. The IPv6 header is only twice as large as the IPv4 header, even though IPv6 addresses are four times as large as IPv4 addresses.

### **Larger Address Space**

IPv6 has 128-bit (16-byte) source and destination IP addresses. Although 128 bits can express over  $3.4 \times 10^{38}$  possible combinations, the large address space of IPv6 has been designed for multiple levels of subnetting and address allocation from the Internet backbone to the individual subnets within an organization.

Even though only a small number of the possible addresses are currently allocated for use by hosts, plenty of addresses are available for future use. With a much larger number of available addresses, address-conservation techniques, such as the deployment of NATs, are no longer necessary.

### **Efficient and Hierarchical Addressing and Routing Infrastructure**

IPv6 global addresses that are used on the IPv6 portion of the Internet are designed to create an efficient, hierarchical, and summarizable routing infrastructure that is based on the common occurrence of multiple levels of Internet service providers.

### **Stateless and Stateful Address Configuration**

To simplify host configuration, IPv6 supports both stateful address configuration (as in the presence of a DHCP server) and stateless address configuration (as in the absence of a DHCP server). With stateless address configuration, hosts on a link automatically configure themselves with IPv6 addresses for the link (called link-local addresses) and with addresses that they derive from prefixes that local routers advertise. Even in the absence of a router, hosts on the same link can configure themselves with link-local addresses and communicate without manual configuration.

### **Built-in Security**

The IPv6 protocol suite requires support for IPSec. This requirement provides a standards-based solution for network security needs and promotes interoperability between different IPv6 implementations.

### **Better Support for QoS**

New fields in the IPv6 header define how traffic is handled and identified. Traffic identification (using a Flow Label field in the IPv6 header) allows routers to identify and provide special handling for packets

## IPv6 Technical Reference (Microsoft Corporation)

belonging to a flow, which is a series of packets between a source and a destination. Because the IPv6 header identifies the traffic, QoS can be supported even when the packet payload is encrypted through IPsec.

### New Protocol for Neighboring Node Interaction

The Neighbor Discovery protocol for IPv6 is a series of Internet Control Message Protocol for IPv6 (ICMPv6) messages that manage the interaction of nodes on the same link (known as neighboring nodes). Neighbor Discovery replaces the broadcast-based Address Resolution Protocol (ARP), ICMPv4 Router Discovery, and ICMPv4 Redirect messages with efficient multicast and unicast Neighbor Discovery messages.

### Extensibility

IPv6 can easily be extended by adding extension headers after the IPv6 header. Unlike options in the IPv4 header, which can support only 40 bytes of options, the size of IPv6 extension headers is constrained only by the size of the IPv6 packet.

### Differences Between IPv4 and IPv6

The following table highlights some of the key differences between IPv4 and IPv6.

**Differences between IPv4 and IPv6**

IPv4	IPv6
Source and destination addresses are 32 bits (4 bytes) in length.	Source and destination addresses are 128 bits (16 bytes) in length. For more information, see "IPv6 Addressing" in "How IPv6 Works."
IPsec support is optional.	IPsec support is required.
IPv4 header does not identify packet flow for QoS handling by routers.	IPv6 header contains Flow Label field, which identifies packet flow for QoS handling by router.
Both routers and the sending host fragment packets.	Only the sending host fragments packets; routers do not.
Header includes a checksum.	Header does not include a checksum.
Header includes options.	All optional data is moved to IPv6 extension headers.
Address Resolution Protocol (ARP) uses broadcast ARP Request frames to resolve an IP address to a link-layer address.	Multicast Neighbor Solicitation messages resolve IP addresses to link-layer addresses. For more information, see "Neighbor Discovery" in "How IPv6 Works."
Internet Group Management Protocol (IGMP) manages membership in local subnet groups.	Multicast Listener Discovery (MLD) messages manage membership in local subnet groups.
ICMP Router Discovery is used to determine the IPv4 address of the best default gateway, and it is optional.	ICMPv6 Router Solicitation and Router Advertisement messages are used to determine the IP address of the best default gateway, and they are required. For more information, see "Neighbor Discovery" in "How IPv6 Works."
Broadcast addresses are used to send traffic to all nodes on a subnet.	IPv6 uses a link-local scope all-nodes multicast address.
Must be configured either manually or	Does not require manual configuration or DHCP.

## IPv6 Technical Reference (Microsoft Corporation)

through DHCP.	
Uses host address (A) resource records in Domain Name System (DNS) to map host names to IPv4 addresses.	Uses host address (AAAA) resource records in DNS to map host names to IPv6 addresses.
Uses pointer (PTR) resource records in the IN-ADDR.ARPA DNS domain to map IPv4 addresses to host names.	Uses pointer (PTR) resource records in the IP6.ARPA DNS domain to map IPv6 addresses to host names.
Must support a 576-byte packet size (possibly fragmented).	Must support a 1280-byte packet size (without fragmentation).

### How IPv6 Works

Internet Protocol version 6 (IPv6) is a networking protocol that allows Windows users to communicate with other users over the Internet. It interacts with Windows naming services such as Domain Name System (DNS) and uses security technologies such as Internet Protocol security (IPSec), because they help facilitate the successful and secure transfer of IP packets between computers.

Ideally, IPv6 is used in a pure environment, that is, an environment where IPv6 is the exclusive Internet protocol used between computers. Currently, however, pure IPv6 transmissions are attainable only with routers that support IPv6 and computers that are running Windows and that support IPv6. As IPv6 supplants IPv4, pure IPv6 across the Internet will become more prevalent and will eventually replace IPv4. Until that occurs, the transition technologies described in this reference can be used to bridge the technological gap between IPv4 and IPv6.

In addition to describing the transition technologies between IPv4 and IPv6, this subject describes how IPv6 relates to other networking protocols, which functions IPv6 performs, how IPv6 addresses are structured and assigned, and how IPv6 packets are structured and routed.

### IPv6 Architecture

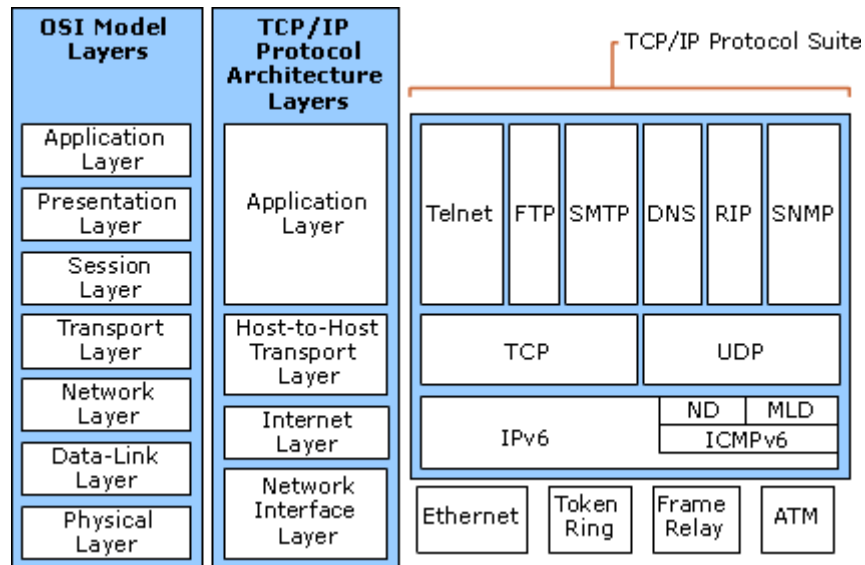
The IPv6 protocol component that is installed in Windows operating systems is a series of interconnected protocols that include Internet Control Message Protocol version 6 (ICMPv6), Multicast Listener Discovery (MLD), and Neighbor Discovery. These core protocols replace the Internet layer protocols in the Defense Advanced Research Projects Agency (DARPA) model. All protocols above the Internet layer rely on the basic services that IPv6 provides. Protocols at the Host-to-Host Transport and Application layers are largely unchanged, except when addresses are part of the payload or part of the data structures that the protocol maintains. For example, both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) must be updated to perform new checksum calculations that include IPv6 addresses. TCP must be updated to store IPv6 addresses in its internal Transmission Control Block (TCB). Routing Information Protocol (RIP) must be updated to send and receive IPv6 route prefixes.

The following figure shows the architecture of the IPv6 core protocols in relation to the Open Systems Interconnection (OSI) model, the TCP/IP protocol architecture, and the other protocols in the TCP/IP suite.

# IPv6 Technical Reference

## (Microsoft Corporation)

### IPv6 Architecture



The following table explains each of the IPv6 core protocols in more detail.

### IPv6 Core Protocols

Protocol	Function
IPv6	IPv6 is a routable protocol that is responsible for the addressing, routing, and fragmenting of packets by the sending host. IPv6 replaces Internet Protocol version 4 (IPv4).
ICMPv6	ICMPv6 is responsible for providing diagnostic functions and reporting errors due to the unsuccessful delivery of IPv6 packets. ICMPv6 replaces ICMPv4.
Neighbor Discovery	Neighbor Discovery is responsible for the interaction of neighboring nodes and includes message exchanges for address resolution, duplicate address detection, router discovery, and router redirects. Neighbor Discovery replaces Address Resolution Protocol (ARP), ICMPv4 Router Discovery, and the ICMPv4 Redirect message.
Multicast Listener Discovery	Multicast Listener Discovery is a series of three ICMPv6 messages that replace version 2 of the Internet Group Management Protocol (IGMP) for IPv4 to manage subnet multicast membership.

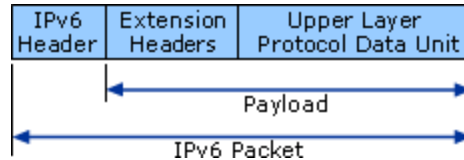
### IPv6

Like IPv4, IPv6 is a connectionless, unreliable datagram protocol that is primarily responsible for addressing and routing packets between hosts. Connectionless means that a session is not established before data is exchanged. Unreliable means that delivery is not guaranteed. IPv6 always makes a best effort attempt to deliver a packet. An IPv6 packet might be lost, delivered out of sequence, duplicated, or delayed. IPv6 does not attempt to recover from these types of errors. The acknowledgment of packets delivered and the recovery of lost packets is the responsibility of an upper layer protocol, such as TCP.

## IPv6 Technical Reference (Microsoft Corporation)

An IPv6 packet consists of an IPv6 header and an IPv6 payload. The IPv6 payload consists of zero or more IPv6 extension headers and an upper layer protocol data unit, such as an ICMPv6 message, a TCP segment, or a UDP message. The following figure shows the structure of an IPv6 packet.

**Structure of an IPv6 Packet**



The following table describes the key fields in the IPv6 header.

**Key Fields in the IPv6 Header**

IP Header Field	Function
Source Address	Identifies the IPv6 address of the original source of the packet.
Destination Address	Identifies the IPv6 address of the intermediate or final destination of the packet.
Next Header	Identifies either the extension header immediately following the IPv6 header or an upper layer protocol, such as ICMPv6, TCP, or UDP.
Hop Limit	Designates the number of subnets, or network segments, on which the packet is allowed to travel before being discarded by a router. The sending host sets the Hop Limit, which prevents packets from endlessly circulating on an IPv6 internetwork. When forwarding an IPv6 packet, routers decrement the Hop Limit by one.

### IPv6 Extension Headers

Zero or more extension headers of varying lengths can be present. If any extension headers are present, a Next Header field in the IPv6 header indicates the first extension header. In a typical IPv6 packet, no extension headers are present. The sending host adds one or more extension headers only if either an intermediate router or the destination requires special handling.

Within each extension header is another Next Header field that indicates the next extension header. The last extension header indicates the upper layer protocol (such as TCP, UDP, or ICMPv6) contained within the upper layer protocol data unit.

The IPv6 header and extension headers replace the existing IPv4 header with options. The new extension header format allows IPv6 to be augmented to support future needs and capabilities. Unlike options in the IPv4 header, IPv6 extension headers have no maximum size and can expand to accommodate all the extension data needed for IPv6 communication.

RFC 2460 defines the following extension headers, which all IPv6 nodes must support:

- Hop-by-Hop Options header
- Destination Options header
- Routing header

# IPv6 Technical Reference

(Microsoft Corporation)

- Fragment header
- Authentication header
- Encapsulating Security Payload header

## Fragmentation in IPv6

If an IPv4 router receives a packet that is too large for the network segment to which the packet is being forwarded and fragmentation of the packet is allowed, IPv4 fragments the original packet into smaller packets that fit on the downstream network segment. In IPv6, only the sending host performs fragmentation. If an IPv6 router cannot forward a packet because it is too large, the router sends an ICMPv6 Packet Too Big message to the sending host and discards the packet.

Use of the Fragment extension header facilitates sending host fragmentation and destination host reassembly.

## ICMPv6

Like IPv4, IPv6 does not provide facilities for reporting errors. Instead, IPv6 uses an updated version of the Internet Control Message Protocol (ICMP) named ICMPv6. ICMPv6 has the common ICMPv4 functions of reporting delivery or forwarding errors and providing a simple echo service for troubleshooting.

The ICMPv6 protocol also provides a framework for the following:

- **Neighbor Discovery:** Neighbor Discovery is a series of five ICMPv6 messages that manage node-to-node communication on a link. Neighbor Discovery replaces Address Resolution Protocol (ARP), ICMPv4 Router Discovery, and the ICMPv4 Redirect message. For more information about Neighbor Discovery, see “Neighbor Discovery” later in this section.
- **Multicast Listener Discovery (MLD):** Multicast Listener Discovery is a series of three ICMP messages that manage subnet multicast membership. Multicast Listener Discovery replaces version 2 of the Internet Group Management Protocol (IGMP) for IPv4. For more information about Multicast Listener Discovery, see “Multicast Listener Discovery” later in this section.

ICMPv6 is required for an IPv6 implementation. The following table lists and describes the ICMPv6 messages that are defined in RFC 2463.

**Common ICMPv6 Messages**

ICMPv6 Message	Function
Echo Request	Sent to check IPv6 connectivity to a particular host.
Echo Reply	Sent in response to an ICMPv6 Echo Request.
Destination Unreachable	Sent by a router or the destination host to inform the sending host that the packet or payload cannot be delivered.
Packet Too Big	Sent by a router to inform a sending host that a packet is too large to forward.
Time Exceeded	Sent by a router to inform a sending host that the Hop Limit of an IPv6 packet has expired.
Parameter Problem	Sent by a router to inform a sending host that an error was encountered in processing the IPv6 header or an IPv6 extension header.

## IPv6 Technical Reference (Microsoft Corporation)

A series of defined Destination Unreachable messages has been defined. The following table lists and describes the most common of these messages.

### Common Destination Unreachable Messages

Destination Unreachable Message	Description
No Route Found	Sent by an IPv6 router when a route to the destination IPv6 address cannot be found.
Communication Prohibited by Administrative Policy	Sent when administrative policy prohibits communication with the destination host. This message is typically sent when a firewall discards a packet.
Destination Address Unreachable	Sent when the destination address is unreachable. This message is typically sent when the destination's link layer address cannot be resolved.
Destination Port Unreachable	Sent when the destination port is unreachable. This message is typically sent when an IPv6 packet containing a UDP message arrived at the destination but no applications were listening on the destination UDP port.

ICMPv6 does not make IPv6 a reliable protocol. ICMPv6 attempts to report errors and to provide feedback on specific conditions. ICMPv6 messages are carried as unacknowledged IPv6 datagrams and are themselves unreliable.

### Path MTU Discovery

The maximum transmission unit (MTU) for a path is the minimum link MTU of all links on a path between a source and a destination. IPv6 packets that are smaller than the path MTU do not require fragmentation by the host and are successfully forwarded by all routers on the path. To discover the path MTU, the sending host uses the receipt of ICMPv6 Packet Too Big messages.

Sending hosts discover the path MTU through the following process:

1. The sending host assumes that the path MTU is the link MTU of the interface on which the traffic is being forwarded.
2. The sending host sends IPv6 datagrams at the path MTU size.
3. If a router on the path is unable to forward the packet over a link because the packet is larger than the link MTU, the router sends an ICMPv6 Packet Too Big message back to the sending host and discards the packet. The Packet Too Big message contains the MTU of the link on which the forwarding failed.
4. The sending host sets the path MTU for packets being sent to the destination to the value of the MTU field in the Packet Too Big message.

The sending host starts again at step 2 and repeats steps 2 through 4 as many times as necessary to discover the path MTU. The sending host determines the path MTU when the node receives either an acknowledgement from the destination or no more Packet Too Big messages.

RFC 1981 recommends that IPv6 nodes support path MTU discovery. Nodes that do not provide this support must use the minimum link MTU of 1280 bytes as the path MTU.

# IPv6 Technical Reference

(Microsoft Corporation)

## Changes in Path MTU

Changes in routing topology can change the path between source and destination. If a path MTU becomes smaller, sending hosts receive Packet Too Big messages, and the nodes must repeat steps 2 through 4 to discover the new path MTU. If a path MTU becomes larger, sending hosts can detect this change (as described in RFC 1981) by attempting to send a larger IPv6 packet after a minimum of 5 minutes (10 minutes are recommended) upon receiving a Packet Too Big message.

## Neighbor Discovery

Neighbor Discovery is a set of ICMPv6 messages and processes that determine relationships between neighboring nodes. Neighbor Discovery replaces ARP, ICMP Router Discovery, and ICMP Redirect used in IPv4, and Neighbor Discovery provides additional functionality.

Hosts use Neighbor Discovery to:

- Discover neighboring routers.
- Discover addresses, address prefixes, and other configuration parameters.

Routers use Neighbor Discovery to:

- Advertise their presence, host configuration parameters, and on-link prefixes.
- Inform hosts of a better next-hop address to forward packets for a specific destination.

Nodes use Neighbor Discovery to:

- Resolve the link-layer address of a neighboring node to which an IPv6 packet is being forwarded.
- Determine when the link-layer address of a neighboring node has changed.
- Determine whether a neighbor is still reachable.

The following table lists and describes the Neighbor Discovery processes that RFC 2461 describes.

### IPv6 Neighbor Discovery Processes

Neighbor Discovery Process	Description
Router discovery	The process by which hosts discover the local routers on attached links. Equivalent to ICMPv4 Router Discovery. For more information, see "Router Discovery" later in this section.
Prefix discovery	The process by which hosts discover the network prefixes for local link destinations. Similar to the ICMPv4 Address Mask Request/Reply. For more information, see "Router Discovery" later in this section.
Parameter discovery	The process by which hosts discover additional operating parameters, including the link MTU and the default hop limit for outgoing packets. For more information, see "Router Discovery" later in this section.
Address autoconfiguration	The process for configuring IP addresses for interfaces in either the presence or absence of a server that provides stateful address configuration using a protocol such as Dynamic Host Configuration Protocol version 6 (DHCPv6). For more information, see "Address Autoconfiguration" later in this section.
Address resolution	The process by which nodes resolve a neighbor's IPv6 address to its link-layer

## IPv6 Technical Reference (Microsoft Corporation)

	address. Equivalent to ARP in IPv4. For more information, see “Address Resolution” later in this section.
Next-hop determination	The process by which nodes determine the IPv6 address of the neighbor to which a packet is being forwarded based on the destination address. The forwarding or next-hop address is either the destination address or the address of an on-link default router. For more information, see “End-to-End IPv6 Delivery Process” later in this section.
Neighbor unreachability detection	The process by which nodes determine that a neighbor is no longer receiving packets. For more information, see “Neighbor Unreachability Detection” later in this section.
Duplicate address detection	The process by which nodes determine whether an address considered for use is already in use by a neighboring node. Equivalent to using gratuitous ARP frames in IPv4. For more information, see “Duplicate Address Detection” later in this section.
Redirect function	The process of informing a host of a better first-hop IPv6 address to reach a destination. Equivalent to the use of the IPv4 ICMP Redirect message. For more information, see “Redirect Function” later in this section.

### Multicast Listener Discovery

Multicast Listener Discovery (MLD) is the IPv6 equivalent of Internet Group Management Protocol version 2 (IGMPv2) for IPv4. MLD is a set of ICMPv6 messages that routers and nodes exchange to enable routers to discover which multicast addresses have listening nodes for each attached interface. As with IGMPv2, MLD discovers only those multicast addresses for which at least one listener exists, not the list of individual multicast listeners for each multicast address. MLD is described in RFC 2710.

In contrast to IGMPv2, MLD uses ICMPv6 messages instead of defining its own message structure. The three types of MLD messages are:

- **Multicast Listener Query:** Routers send Multicast Listener Query messages to query a link for multicast listeners. Multicast Listener Query messages are either General Query messages or Multicast-Address-Specific Query messages. Routers send General Query messages to query for multicast listeners of all multicast addresses. Routers send Multicast-Address-Specific Query messages to query for multicast listeners of a specific multicast address. The two message types are distinguished by the multicast destination address in the IPv6 header and a multicast address within the Multicast Listener Query message.
- **Multicast Listener Report:** Multicast listeners send Multicast Listener Report messages to either report interest in receiving multicast traffic for a specific multicast address or to respond to a Multicast Listener Query message.
- **Multicast Listener Done:** Multicast listeners send Multicast Listener Done messages to report that they are no longer interested in receiving multicast traffic for a specific multicast address.

An MLD message packet consists of an IPv6 header, a Hop-by-Hop Options extension header, and the MLD message. The Hop-by-Hop Options extension header contains the IPv6 Router Alert Option that RFC 2711 describes. This option ensures that routers process MLD messages sent to multicast addresses on which the routers are not listening.

# IPv6 Technical Reference

(Microsoft Corporation)

## IPv6 Application Interfaces

As in IPv4, applications can use the services of IPv6 by using the Windows Sockets application programming interface (API). A socket is defined by a protocol and an address on the host. The format of the address is specific to each protocol. For TCP and UDP traffic over IPv6, the address is the combination of the IPv6 address and port. Two sockets, one for each end of the connection, form a bi-directional communications path.

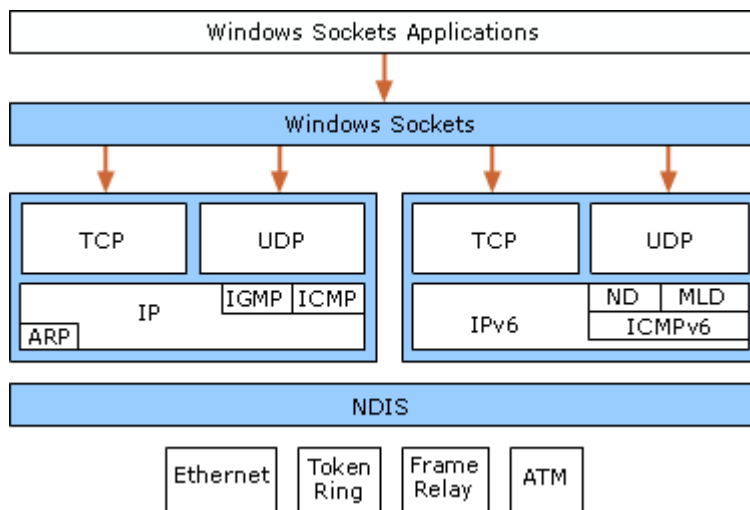
To communicate with IPv6, an application can use Windows Sockets to specify the protocol (IPv6), the IPv6 address of the destination host, and the port of the destination application. After the application is connected, information can be sent and received.

Until IPv6 is broadly adopted, most hosts will be IPv6/IPv4 hosts, using implementations of both the IPv6 and IPv4 protocols. Developers could modify current applications to test for and then selectively use either IPv4 or IPv6 as their transport for data. However, it makes more sense to modify the applications to use a single Windows Sockets function to perform network operations regardless of whether IPv4, IPv6, or both are installed.

A good example of making applications IP version-independent is to replace all instances of the Windows Sockets function `gethostbyname()` with the Windows Sockets function `getaddrinfo()`. The `gethostbyname()` function is commonly used to resolve a specified name to an IPv4 address and never returns an IPv6 address. The function `getaddrinfo()` resolves names to both IPv4 and IPv6 addresses, depending on whether IPv4 and IPv6 is installed and which types of addresses the Domain Name System (DNS) name query returns. The application need not test for the existence of both protocols or both types of addresses in the response. By replacing `gethostbyname()` with `getaddrinfo()`, the developer makes the application IP version-independent for name resolution.

Windows Sockets has been extended to support IPv6 either directly, using IPv6-specific Windows Sockets functions and structures, or indirectly, using IP version-independent functions. The following figure shows the Windows Sockets architecture for Windows Server 2003.

**Windows Sockets for Both IPv4 and IPv6**



## IPv6 Addressing

The most obvious distinguishing feature of IPv6 is its use of much larger addresses. The size of an address in IPv6 is 128 bits, which is four times larger than an IPv4 address. A 32-bit address space

## IPv6 Technical Reference (Microsoft Corporation)

includes 232 or 4,294,967,296 possible addresses. A 128-bit address space includes 2128 or 340,282,366,920,938,463,463,374,607,431,768,211,456 (or  $3.4 \times 10^{38}$ ) possible addresses.

The IPv4 address space was designed in the late 1970s, and it seemed impossible to exhaust. However, addresses were not allocated in a way that anticipated changes in technology and an explosion in the number of hosts on the Internet. The IPv4 address space was consumed to the point that, by 1992, it clearly needed a replacement.

It is even harder to conceive that the IPv6 address space will be consumed. A 128-bit address space provides 655,570,793,348,866,943,898,599 ( $6.5 \times 10^{23}$ ) addresses for every square meter of the Earth's surface.

It is important to remember that the decision to make the IPv6 address 128 bits long was not so that every square meter of the Earth could have  $6.5 \times 10^{23}$  addresses. Rather, the relatively large size of the IPv6 address is designed to be subdivided into hierarchical routing domains that reflect the topology of the modern Internet. The use of 128 bits allows multiple levels of hierarchy and flexibility in designing hierarchical addressing and routing that is currently lacking in the IPv4-based Internet.

RFC 3513 describes the IPv6 addressing architecture.

### IPv6 Address Syntax

IPv4 addresses are represented in dotted-decimal format. These 32-bit addresses are divided along 8-bit boundaries. Each set of 8 bits is converted to its decimal equivalent and separated from the other sets by periods. For IPv6, the 128-bit address is divided along 16-bit boundaries. Each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is known as colon-hexadecimal.

The following is an IPv6 address in binary form:

```
0010000111011010000000000110100110000000000000000010111100111011  
0000001010101010000000001111111111111110001010001001110001011010
```

The 128-bit address is divided along 16-bit boundaries:

```
0010000111011010    0000000011010011    0000000000000000    0010111100111011  
0000001010101010    0000000011111111    1111111000101000    1001110001011010
```

Each 16-bit block is converted to hexadecimal and delimited with colons. The result is:

```
21DA:00D3:0000:2F3B:02AA:00FF:FE28:9C5A
```

IPv6 representation can be further simplified by removing the leading zeros within each 16-bit block. However, each block must have at least a single digit. With leading zero suppression, the address representation becomes:

```
21DA:D3:0:2F3B:2AA:FF:FE28:9C5A
```

### Compressing Zeros

Some types of addresses contain long sequences of zeros. To further simplify the representation of IPv6 addresses, a contiguous sequence of 16-bit blocks set to 0 in the colon-hexadecimal format can be compressed to "::," known as double-colon.

## IPv6 Technical Reference (Microsoft Corporation)

For example, the link-local address of FE80:0:0:0:2AA:FF:FE9A:4CA2 can be compressed to FE80::2AA:FF:FE9A:4CA2. The multicast address FF02:0:0:0:0:0:0:2 can be compressed to FF02::2.

Zero compression can be used to compress only a single contiguous series of 16-bit blocks expressed in colon-hexadecimal notation. You cannot use zero compression to include part of a 16-bit block. For example, you cannot express FF02:30:0:0:0:0:0:5 as FF02:3::5.

To determine how many 0 bits are represented by the double colon, you can count the number of blocks in the compressed address, subtract this number from 8, and then multiply the result by 16. For example, the address FF02::2 has two blocks (the “FF02” block and the “2” block.) The number of 0 bits expressed by the double colon is 96 ( $96 = (8 - 2) \times 16$ ).

Zero compression can be used only once in a given address. Otherwise, you could not determine the number of 0 bits represented by each double colon.

### IPv6 Prefixes

The prefix is the part of the address that indicates which bits have fixed values or reflect the subnet identifier. Prefixes for IPv6 subnet identifiers and routes are expressed in the same way as Classless Inter-Domain Routing (CIDR) notation for IPv4, that is, in address/prefix-length notation. For example, 21DA:D3::/48 is a route prefix and 21DA:D3:0:2F3B::/64 is a subnet prefix.

**Note:** IPv4 implementations commonly use a dotted decimal representation of the network prefix known as the subnet mask. IPv6 does not support subnet masks. IPv6 supports only the prefix-length notation.

### Types of IPv6 Addresses

IPv6 supports three types of addresses:

- **Unicast:** A unicast address identifies a single interface within the scope of the type of unicast address. With the appropriate unicast routing topology, packets addressed to a unicast address are delivered to a single interface. To accommodate load-balancing systems, RFC 3513 allows multiple interfaces to use the same address as long as they appear as a single interface to the IPv6 implementation on the host.
- **Multicast:** A multicast address identifies multiple interfaces. With the appropriate multicast routing topology, packets addressed to a multicast address are delivered to all interfaces that are identified by the address. A multicast address is used for one-to-many communication, with delivery to multiple interfaces.
- **Anycast:** An anycast address identifies multiple interfaces. With the appropriate routing topology, packets addressed to an anycast address are delivered to a single interface, the nearest interface that is identified by the address. The nearest interface is defined as being closest in terms of routing distance. An anycast address is used for one-to-one-of-many communication, with delivery to a single interface.

IPv6 addresses always identify interfaces, not nodes. A node is identified by any unicast address that is assigned to one of its interfaces.

# IPv6 Technical Reference

(Microsoft Corporation)

**Note:** RFC 3513 does not define a broadcast address. All types of IPv4 broadcast addressing are performed in IPv6 using multicast addresses. For example, the subnet and limited broadcast addresses from IPv4 are replaced with the link-local scope all-nodes multicast address of FF02::1.

## Unicast IPv6 Addresses

Unicast IPv6 addresses fall into one of five types:

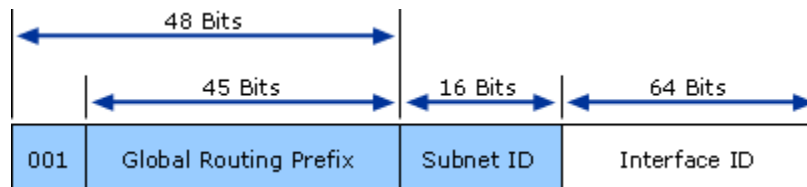
- Global unicast addresses
- Link-local addresses
- Site-local addresses
- Special addresses
- Compatibility addresses

Global unicast addresses are equivalent to public IPv4 addresses. They are globally routable and reachable on the IPv6 Internet.

Unlike the current IPv4-based Internet, which is a mixture of both flat and hierarchical routing, the IPv6-based Internet has been designed from its foundation to support efficient, hierarchical addressing and routing. The scope (that is, the region of the IPv6 internetwork over which the address is unique) of a global unicast address is the entire IPv6 Internet.

The following figure shows the structure of a global unicast address as defined in RFC 3587.

**The Global Unicast Address**



Global unicast addresses contain four fields:

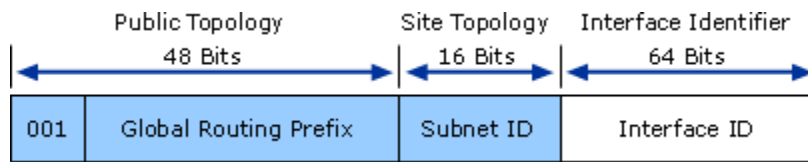
- The three high-order bits are set to 001. The address prefix for currently assigned global addresses is 2000::/3.
- The Global Routing Prefix indicates the global routing prefix for a specific organization's site. The combination of the three fixed bits and the 45-bit Global Routing Prefix creates a 48-bit site prefix, which is assigned to an individual site of an organization. After this prefix is assigned, routers on the IPv6 Internet forward IPv6 traffic matching the 48-bit prefix to the routers of the organization's site.
- The Subnet ID is used within an organization's site to identify subnets. This field is 16 bits long. The organization's site can use these 16 bits within its site to create 65,536 subnets or multiple levels of addressing hierarchy and an efficient routing infrastructure.
- The Interface ID indicates the interface on a specific subnet within the site. This field is 64 bits long.

The following figure shows how fields within the global unicast address create a three-level structure.

# IPv6 Technical Reference

(Microsoft Corporation)

## The Three-level Structure of the Global Unicast Address



The public topology is the collection of larger and smaller ISPs that provide access to the IPv6 Internet. The site topology is the collection of subnets within an organization's site. The interface identifier identifies a specific interface on a subnet within an organization's site. For more information about global unicast addresses, see RFC 3587 in the IETF RFC Database.

### Local-Use Unicast Addresses

Local-use unicast addresses fall into one of two types:

- Link-local addresses are used between on-link neighbors and for Neighbor Discovery processes.
- Site-local addresses are used between nodes in the same site.

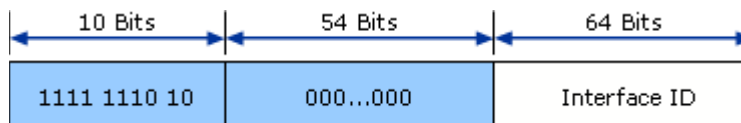
### Link-local Addresses

Nodes use link-local addresses when communicating with neighboring nodes on the same link. For example, on a single-link IPv6 network with no router, hosts use link-local addresses to communicate with other hosts on the link. Link-local addresses are equivalent to Automatic Private IP Addressing (APIPA) IPv4 addresses autoconfigured on computers that are running Windows. APIPA addresses use the 169.254.0.0/16 prefix. The scope of a link-local address is the local link.

A link-local address is required for Neighbor Discovery processes and is always automatically configured, even in the absence of all other unicast addresses. For more information about how link-local addresses are configured, see "Address Autoconfiguration" later in this section.

The following figure shows the structure of the link-local address.

### The Link-Local Address



Link-local addresses always begin with FE80. With the 64-bit interface identifier, the prefix for link-local addresses is always FE80::/64. An IPv6 router never forwards link-local traffic beyond the link.

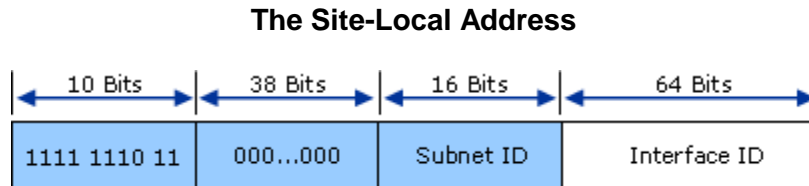
### Site-local Addresses

Site-local addresses are equivalent to the IPv4 private address space (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16). For example, private intranets that do not have a direct, routed connection to the IPv6 Internet can use site-local addresses without conflicting with global addresses. Site-local addresses are not reachable from other sites, and routers must not forward site-local traffic outside the site. Site-local addresses can be used in addition to global addresses. The scope of a site-local address is the site (the organization internetwork).

## IPv6 Technical Reference (Microsoft Corporation)

Unlike link-local addresses, site-local addresses are not automatically configured and must be assigned through either stateless or stateful address configuration.

The following figure shows the structure of the site-local address.



The first 10 bits are always fixed for site-local addresses, beginning with FEC0::/10. After the 10 fixed bits is a 54-bit subnet identifier (Subnet ID field) that provides 54 bits with which you can create a hierarchical and summarizable routing infrastructure within the site. After the Subnet ID field is a 64-bit Interface ID field that identifies a specific interface on a subnet.

### Special IPv6 Addresses

The following are special IPv6 addresses:

- **Unspecified Address:** The unspecified address (0:0:0:0:0:0:0:0 or ::) indicates the absence of an address. It is equivalent to the IPv4 unspecified address of 0.0.0.0. The unspecified address is typically used as a source address for packets attempting to verify the uniqueness of a tentative address. The unspecified address is never assigned to an interface or used as a destination address.
- **Loopback Address:** The loopback address (0:0:0:0:0:0:0:1 or ::1) is used to identify a loopback interface, enabling a node to send packets to itself. It is equivalent to the IPv4 loopback address of 127.0.0.1. Packets addressed to the loopback address must never be sent on a link or forwarded by a router.

### Compatibility Addresses

To aid in the migration from IPv4 to IPv6 and the coexistence of both types of hosts, the following addresses are defined:

- **IPv4-compatible Address:** The IPv4-compatible address, 0:0:0:0:0:w.x.y.z or ::w.x.y.z (where w.x.y.z is the dotted decimal representation of a public IPv4 address), is used by IPv6/IPv4 nodes that are communicating using IPv6. IPv6/IPv4 nodes support both IPv4 and IPv6 protocols. When an IPv4-compatible address is used as an IPv6 destination, the IPv6 traffic is automatically encapsulated with an IPv4 header and sent to the destination using the IPv4 infrastructure.
- **IPv4-mapped Address:** The IPv4-mapped address, 0:0:0:0:FFFF:w.x.y.z or ::FFFF:w.x.y.z, is used to represent an IPv4-only node to an IPv6 node. It is used only for internal representation. The IPv4-mapped address is never used as a source or destination address of an IPv6 packet.
- **6to4 Address:** The 6to4 address is used for communicating between two nodes running both IPv4 and IPv6 over an IPv4 routing infrastructure. The 6to4 address is formed by combining the prefix 2002::/16 with the 32 bits of a public IPv4 address of the node, forming a 48-bit prefix. 6to4 is a tunneling technique described in RFC 3056.

## IPv6 Technical Reference (Microsoft Corporation)

- **ISATAP Address:** The Internet draft titled “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)” defines ISATAP addresses used between two nodes running both IPv4 and IPv6 over an IPv4 routing infrastructure. ISATAP addresses use the locally administered interface ID `::0:5EFE:w.x.y.z` where `w.x.y.z` is any unicast IPv4 address, which includes both public and private addresses.

The ISATAP interface ID can be combined with any 64-bit prefix that is valid for IPv6 unicast addresses. This includes the link-local address prefix (`FE80::/64`), site-local prefixes, and global prefixes.

- **Teredo Address:** Teredo addresses use the prefix `3FFE:831F::/32`. Beyond the first 32 bits, Teredo addresses are used to encode the IPv4 address of a Teredo server, flags, and the encoded version of the external address and port of a Teredo client. An example of a Teredo address is `3FFE:831F:CE49:7601:8000:FFFF:62C3:FFFE`. Teredo addresses are used to represent a host when using the automatic tunneling mechanism defined in the Internet draft titled “Teredo: Tunneling IPv6 over UDP through NATs.” For more information, see “Teredo” later in this section.

### Multicast IPv6 Addresses

In IPv6, multicast traffic operates in the same way that it does in IPv4. Arbitrarily located IPv6 nodes can listen for multicast traffic on arbitrary IPv6 multicast addresses. IPv6 nodes can listen to multiple multicast addresses at the same time. Nodes can join or leave a multicast group at any time.

IPv6 multicast addresses have the first 8 bits set to `1111 1111`. An IPv6 address is easy to classify as multicast because it always begins with “FF.” Multicast addresses cannot be used as source addresses or as intermediate destinations in a Routing header.

Beyond the first 8 bits, multicast addresses include additional structure to identify their flags, scope, and multicast group. The following figure shows the structure of the IPv6 multicast address.

**The IPv6 Multicast Address**



The fields in the multicast address are:

- **Flags:** Indicates flags set on the multicast address. The size of this field is 4 bits. As of RFC 3513, the only flag defined is the Transient (T) flag. The T flag uses the low-order bit of the Flags field. When set to 0, the T flag indicates that the multicast address is a well-known multicast address that has been permanently assigned by the Internet Assigned Numbers Authority (IANA). When set to 1, the T flag indicates that the multicast address is a transient multicast address that IANA has not permanently assigned.
- **Scope:** Indicates the scope of the IPv6 internetwork for which the multicast traffic is intended. The size of this field is 4 bits. In addition to using information provided by multicast routing protocols, routers use the multicast scope to determine whether they can forward multicast traffic. The most prevalent values for the Scope field are 1 (interface-local scope), 2 (link-local scope), and 5 (site-

## IPv6 Technical Reference (Microsoft Corporation)

local scope). For example, traffic with the multicast address of FF02::2 has a link-local scope. An IPv6 router never forwards this traffic beyond the local link.

- **Group ID:** Identifies the multicast group and is unique within the scope. The size of this field is 112 bits. Permanently assigned group IDs are independent of the scope. Transient group IDs are relevant only to a specific scope. Multicast addresses from FF01:: through FF0F:: are reserved, well-known addresses.

To identify all nodes for the interface-local and link-local scopes, the following addresses are defined:

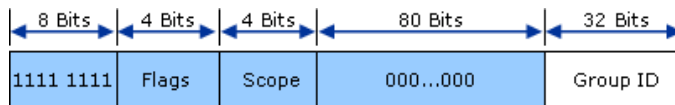
- FF01::1 (interface-local scope all-nodes multicast address)
- FF02::1 (link-local scope all-nodes multicast address)

To identify all routers for the interface-local, link-local, and site-local scopes, the following addresses are defined:

- FF01::2 (interface-local scope all-routers multicast address)
- FF02::2 (link-local scope all-routers multicast address)
- FF05::2 (site-local scope all-routers multicast address)

With 112 bits for the group ID, it is possible to have 2<sup>112</sup> group IDs. However, because of the way in which IPv6 multicast addresses are mapped to Ethernet multicast media access control (MAC) addresses, RFC 3513 recommends assigning the group ID from the low-order 32 bits of the IPv6 multicast address and setting the remaining original group ID bits to 0. By using only the low-order 32 bits, each group ID maps to a unique Ethernet multicast MAC address. The following figure shows the recommended IPv6 multicast address.

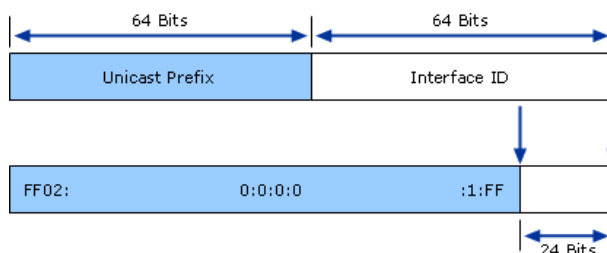
### The Recommended IPv6 Multicast Address Using a 32-bit Group ID



### Solicited-node Address

The solicited-node address facilitates the efficient querying of network nodes during address resolution. In IPv4, the ARP Request frame is sent to the MAC-level broadcast, disturbing all nodes on the network segment, including those that are not running IPv4. IPv6 uses the Neighbor Solicitation message to perform address resolution. However, instead of using the local-link scope all-nodes multicast address as the Neighbor Solicitation message destination, which would disturb all IPv6 nodes on the local link, the solicited-node multicast address is used. The following figure shows how the solicited-node multicast address comprises the prefix FF02::1:FF00:0/104 and the last 24-bits of the IPv6 address that is being resolved.

### Mapping Unicast IPv6 Addresses to Solicited Node IPv6 Addresses



## IPv6 Technical Reference (Microsoft Corporation)

For example, for the node with the link-local IPv6 address of FE80::2AA:FF:FE28:9C5A, the corresponding solicited-node address is FF02::1:FF28:9C5A. This node is listening for multicast traffic at the solicited-node address of FF02::1:FF28:9C5A and, for interfaces that correspond to a physical network adapter, has registered the corresponding multicast address with the network adapter. To resolve the address of FE80::2AA:FF:FE28:9C5A to its link-layer address, a neighboring node sends a Neighbor Solicitation to the solicited-node address of FF02::1:FF28:9C5A.

The result of using the solicited-node multicast address is that address resolutions, a common occurrence on a link, are not required to use a mechanism that disturbs all network nodes. By using the solicited-node address, address resolution disturbs very few nodes. In practice, due to the relationship between the Ethernet MAC address, the IPv6 interface ID, and the solicited-node address, the solicited-node address acts as a pseudo-unicast address for very efficient address resolution.

### **Anycast IPv6 Addresses**

An anycast address is assigned to multiple interfaces. The routing infrastructure forwards packets that are addressed to an anycast address to the nearest interface to which the anycast address is assigned. To facilitate delivery, the routing infrastructure must track the interfaces that have been assigned anycast addresses and their distance in terms of routing metrics. At present, anycast addresses are used only as destination addresses and are assigned only to routers. Anycast addresses are assigned out of the unicast address space, and the scope of an anycast address is the scope of the type of unicast address from which the anycast address is assigned.

The Subnet-Router anycast address is predefined and required. It is created from the subnet prefix for a given interface. To construct the Subnet-Router anycast address, the bits in the subnet prefix are fixed at their appropriate values, and the remaining bits are set to 0. All router interfaces attached to a subnet are assigned the Subnet-Router anycast address for that subnet. The Subnet-Router anycast address is used to communicate with one of multiple routers attached to a remote subnet.

### **IPv6 Addresses for a Host**

An IPv4 host with a single network adapter typically has a single IPv4 address assigned to that adapter. An IPv6 host, however, usually has multiple IPv6 addresses — even with a single interface. An IPv6 host is assigned the following unicast addresses:

- A link-local address for each interface.
- Unicast addresses for each interface (which could be a site-local address and one or multiple global unicast addresses).
- The loopback address (::1) for the loopback interface.

Typical IPv6 hosts are logically multi-homed because they have at least two addresses with which they can receive packets — a link-local address for local link traffic and a routable site-local or global address.

Additionally, each host is listening for traffic on the following multicast addresses:

- The interface-local scope all-nodes multicast address (FF01::1).
- The link-local scope all-nodes multicast address (FF02::1).
- The solicited-node address for each unicast address on each interface.

## IPv6 Technical Reference (Microsoft Corporation)

- The multicast addresses of joined groups on each interface.

### IPv6 Addresses for a Router

An IPv6 router is assigned the following unicast addresses:

- A link-local address for each interface.
- Unicast addresses for each interface (which could be a site-local address and one or multiple global unicast addresses).
- A Subnet-Router anycast address.
- Additional anycast addresses (optional).
- The loopback address (::1) for the loopback interface.

Additionally, each router is listening for traffic on the following multicast addresses:

- The interface-local scope all-nodes multicast address (FF01::1).
- The interface-local scope all-routers multicast address (FF01::2).
- The link-local scope all-nodes multicast address (FF02::1).
- The link-local scope all-routers multicast address (FF02::2).
- The site-local scope all-routers multicast address (FF05::2).
- The solicited-node address for each unicast address on each interface.
- The multicast addresses of joined groups on each interface.

### IPv6 Interface Identifiers

The last 64 bits of an IPv6 address are the interface identifier that is unique to the 64-bit prefix of the IPv6 address. IPv6 interface identifiers are determined as follows:

- Derived from the Extended Unique Identifier (EUI)-64 address.
- Generated randomly and changed over time to provide a level of anonymity.
- Assigned during stateful address autoconfiguration (for example, through DHCPv6).

### EUI-64 Address-based Interface Identifiers

RFC 3513 states that all unicast addresses that use the prefixes 001 through 111 must also use a 64-bit interface identifier derived from the EUI-64 address. EUI-64 addresses have 64-bits and are defined by the Institute of Electrical and Electronic Engineers (IEEE). EUI-64 addresses are either assigned to network adapters or derived from IEEE 802 addresses.

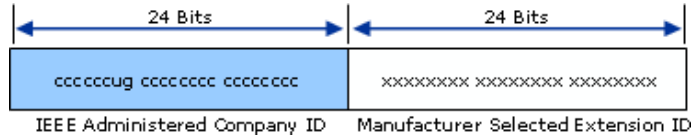
### IEEE 802 Addresses

Traditional interface identifiers for network adapters use a 48-bit address called an IEEE 802 address. It consists of a 24-bit company ID (also called the manufacturer ID) and a 24-bit extension ID (also called the board ID). The combination of the company ID, which is uniquely assigned to each manufacturer of network adapters, and the board ID, which is uniquely assigned to each network adapter at the time of assembly, produces a globally unique 48-bit address. This 48-bit address is also called the physical, hardware, or media access control (MAC) address.

The following figure shows the structure of the 48-bit IEEE 802 address.

# IPv6 Technical Reference (Microsoft Corporation)

## 48-bit IEEE 802 Address



The IEEE 802 address includes the following defined bits:

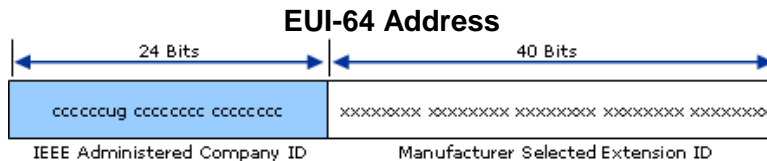
- **Universal/Local (U/L):** The next-to-the-low order bit in the first byte indicates whether the address is universally or locally administered. If the U/L bit is set to 0, the IEEE (through the designation of a unique company ID) has administered the address. If the U/L bit is set to 1, the address is locally administered, which means that the network administrator has overridden the manufactured address and specified a different address.
- **Individual/Group (I/G):** The low-order bit of the first byte indicates whether the address is an individual address (unicast) or a group address (multicast). If the I/G bit is set to 0, the address is a unicast address. If the I/G bit is set to 1, the address is a multicast address.

For a typical IEEE 802 network adapter address, both the U/L and I/G bits are set to 0, indicating a universally administered, unicast MAC address.

## IEEE EUI-64 Addresses

The IEEE Extended Unique Identifier (EUI)-64 address represents a newer standard for network interface addressing. The company ID is still 24 bits, but the extension ID is 40 bits, creating a much larger address space for each network adapter manufacturer. The EUI-64 address uses the U/L and I/G bits in the same way as the IEEE 802 address.

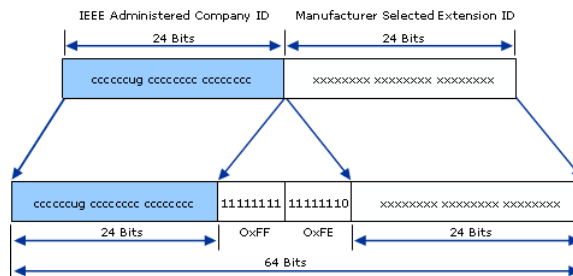
The following figure shows the structure of the EUI-64 address.



## Mapping IEEE 802 Addresses to EUI-64 Addresses

To create an EUI-64 address from an IEEE 802 address, the 16 bits of 11111111 11111110 (0xFFFFE) are inserted into the IEEE 802 address between the company ID and the extension ID. The following figure shows an example of such a conversion.

### Conversion of an IEEE 802 Address to an EUI-64 Address



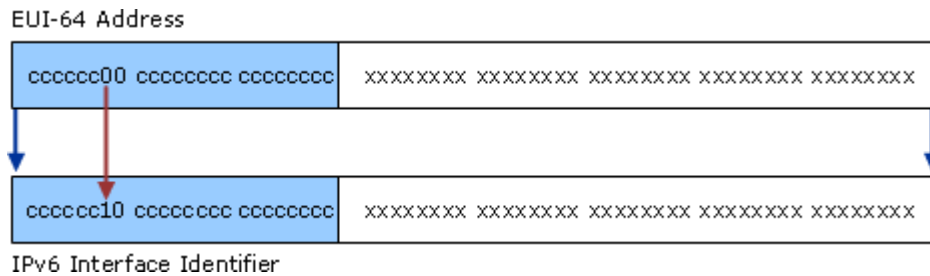
# IPv6 Technical Reference

## (Microsoft Corporation)

### Mapping EUI-64 Addresses to IPv6 Interface Identifiers

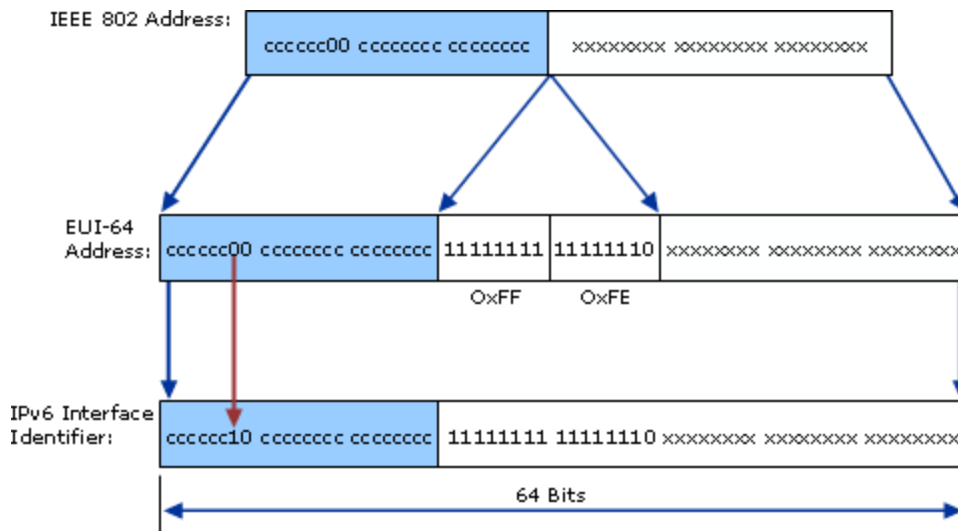
To obtain the 64-bit interface identifier for an IPv6 unicast address, the U/L bit in the EUI-64 address is complemented. (If it is set to 1, it is changed to 0, and if it is set to 0, it is changed to 1.) The following figure shows the conversion for a universally administered, unicast EUI-64 address.

#### Conversion of a Universally Administered, Unicast EUI-64 Address to an IPv6 Interface Identifier



To obtain an IPv6 interface identifier from an IEEE 802 address, you must first map the IEEE 802 address to an EUI-64 address, and then complement the U/L bit. The following figure shows this conversion process for a universally administered, unicast IEEE 802 address.

#### Conversion of a Universally Administered, Unicast IEEE 802 Address to an IPv6 Interface Identifier



### IEEE 802 address Conversion Example

Host A has the Ethernet MAC address of 00-AA-00-3F-2A-1C. First, it is converted to EUI-64 format by inserting FF-FE between the third and fourth bytes, yielding 00-AA-00-FF-FE-3F-2A-1C. The U/L bit, which is the seventh bit in the first byte, is then complemented. Before the conversion, the first byte in binary form is 00000000. When the seventh bit is complemented, it becomes 00000010 (0x02). The final result is 02-AA-00-FF-FE-3F-2A-1C which, when converted to colon-hexadecimal notation, becomes the interface identifier 2AA:FF:FE3F:2A1C. As a result, the link-local address that corresponds to the network adapter with the MAC address of 00-AA-00-2A-1C is FE80::2AA:FF:FE3F:2A1C.

# IPv6 Technical Reference

## (Microsoft Corporation)

**Note:** When complementing the U/L bit, add 0x2 to the first byte if the address is universally administered, and subtract 0x2 from the first byte if the address is locally administered.

### Temporary Address Interface Identifiers

In today's Internet, a typical Internet user connects to an Internet service provider (ISP) and obtains an IPv4 address using the Point-to-Point Protocol (PPP) and the Internet Protocol Control Protocol (IPCP). Each time the user connects, a different IPv4 address might be obtained. Because of this, it is difficult to track a dial-up user's traffic on the Internet on the basis of IP address.

For dial-up connections that are based on IPv6, the user is assigned a 64-bit prefix through router discovery and stateless address autoconfiguration after the connection is made. If the interface identifier is always based on the EUI-64 address (as derived from the static IEEE 802 address), it is possible to identify the traffic of a specific node regardless of the prefix, making it easy to track a specific user and their use of the Internet. To address this concern and provide a level of anonymity, RFC 3041 describes an alternative IPv6 interface identifier that is randomly generated and changes over time.

The initial interface identifier is generated by using random numbers. For computers that cannot store any historical information for generating future interface identifier values, a different random interface identifier is generated each time the IPv6 protocol is initialized. For IPv6 systems that have storage capabilities, a history value is stored. When the IPv6 protocol is initialized, a new interface identifier is generated through the following process:

1. Retrieve the history value from storage, and append the interface identifier based on the EUI-64 address of the adapter.
2. Compute a Message Digest-5 (MD5) one-way encryption hash over the quantity in step 1.
3. Save the last 64 bits of the MD5 hash computed in step 2 as the history value for the next interface identifier computation.
4. Take the first 64 bits of the MD5 hash computed in Step 2, and set the seventh bit to 0. The seventh bit corresponds to the U/L bit which, when set to 0, indicates a locally administered IPv6 interface identifier. The result is the IPv6 interface identifier.

The resulting IPv6 address, based on this random interface identifier, is known as a temporary or anonymous address. Temporary addresses are generated for public address prefixes that use stateless address autoconfiguration. Temporary addresses are used for the shorter of the valid and preferred lifetimes:

- The lifetimes included in the Prefix Information option in the received Router Advertisement message.
- Local default values of one week for valid lifetime and one day for preferred lifetime.

After the valid lifetime of the temporary address expires, a new interface identifier and temporary address is generated.

# IPv6 Technical Reference

(Microsoft Corporation)

## Subnetting Global Address Prefixes

For most users of IPv6, subnetting the IPv6 address space consists of using subnetting techniques to divide the Subnet ID portion of the global address. This approach allows for route summarization and delegation of the remaining address space to different portions of an IPv6 intranet.

For the global address, the first 48 bits are fixed and allocated by an ISP.

Subnetting the subnet ID portion of a global address space requires two steps:

1. Determine the number of bits to be used for the subnetting.
2. Enumerate the new subnetted network prefixes.

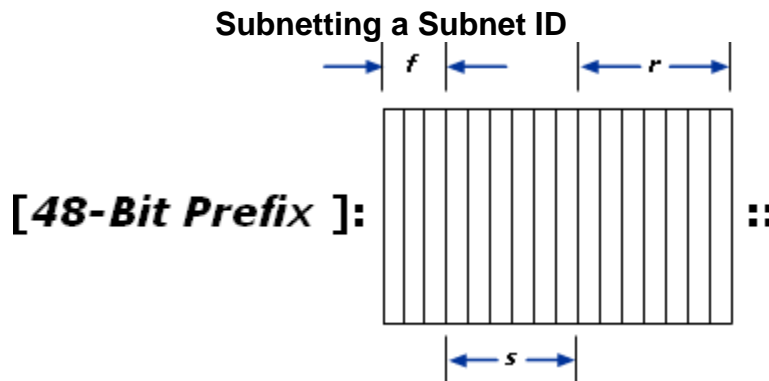
You are not required to subnet in any specific fashion. The subnetting technique described here assumes that subnetting is done by dividing the 16-bit address space of the subnet ID using the high-order bits in the subnet ID. Although this method promotes hierarchical addressing and routing, it is not required. For example, in a small organization with a small number of subnets, you can also create a flat addressing space for the subnet ID by numbering the subnets from 0 to 65,535.

### Step 1: Determining the Number of Subnetting Bits

The number of bits being used for subnetting determines the possible number of new subnetted network prefixes that can be allocated to portions of your network based on geographical or departmental divisions. In a hierarchical routing infrastructure, you need to determine how many network prefixes, and therefore how many bits, are needed at each level in the hierarchy. The more bits you choose for the various levels of the hierarchy, the fewer bits you have to enumerate individual subnets in the last level of the hierarchy.

For example, a network administrator can decide to implement a two-level hierarchy reflecting a geographical/departmental structure, use 4 bits for the geographical level, and use 6 bits for the departmental level. With this approach, each department in each geographical location has only 6 bits of subnetting space left ( $16 - 6 - 4$ ) or only 64 ( $=2^6$ ) subnets per department.

On any given level in the hierarchy, some number of bits is already fixed by the next level up in the hierarchy ( $f$ ), some number of bits is used for subnetting at the current level in the hierarchy ( $s$ ), and some number of bits remain for the next level down in the hierarchy ( $r$ ). At all times,  $f+s+r = 16$ . The following figure shows this relationship.



### Step 2: Enumerating Subnetted Network Prefixes

Based on the number of bits used for subnetting, you must list the new subnetted network prefixes.

## IPv6 Technical Reference (Microsoft Corporation)

There are two main approaches:

- **Hexadecimal:** Enumerate new subnetted network prefixes by using hexadecimal representations of the subnet ID and increment.
- **Decimal:** Enumerate new subnetted network prefixes by using decimal representations of the subnet ID and increment.

Either method produces the same result: an enumerated list of subnetted network prefixes.

To create the enumerated list of subnetted network prefixes using the hexadecimal method:

1. Based on  $s$ , the number of bits chosen for subnetting, and  $m$ , the length of the network prefix being subnetted, calculate the following:

$$f = m - 48$$

$f$  is the number of bits within the global address prefix that are already fixed.

$$n = 2^s$$

$n$  is the number of network prefixes that are obtained.

$$i = 2^{16-(f+s)}$$

$i$  is the incremental value between each successive subnet ID expressed in hexadecimal.

$$l = 48+f+s$$

$l$  is the length of the new subnetted network prefixes.

2. Create a two-column table with  $n$  entries. The first column is the network prefix number (starting with 1), and the second column is the new subnetted network prefix.
3. In the first table entry, the subnetted network prefix is the original network prefix with the new prefix length. For example, based on  $F$ , the hexadecimal value of the subnet ID being subnetted, the subnetted network prefix is [48-bit prefix]: $F::/l$ .
4. In the next table entry, increase the value within the subnet ID portion of the site-local or global address by  $i$ . For example, in the second table entry, the subnetted prefix is [48-bit prefix]: $F+i::/l$ .
5. Repeat step 4 until the table is complete.

For example, to perform a 3-bit subnetting of the site-local network prefix  $3FFE:FFFF:0:C000::/51$ , first calculate the values of the number of prefixes, the increment, and the new prefix length. The starting values are  $F=0xC000$ ,  $s=3$ , and  $f=51-48=3$ . The number of prefixes is 8 ( $n=2^3$ ). The increment is  $0x400$  ( $i=2^{16-(3+3)}=1024=0x400$ ). The new prefix length is 54 ( $l=48+3+3$ ).

Next, construct a table with 8 entries. The entry for the network prefix 1 is  $3FFE:FFFF:0:C000::/54$ . Additional entries in the table are successive increments of  $i$  in the subnet ID portion of the network prefix, as shown in the following table.

**IPv6 Technical Reference**  
(Microsoft Corporation)

## IPv6 Technical Reference (Microsoft Corporation)

### Hexadecimal Subnetting Technique for Network Prefix 3FFE:FFFF:0:C000::/51

Network Prefix	Subnetted Network Prefix
1	3FFE:FFFF:0:C000::/54
2	3FFE:FFFF:0:C400::/54
3	3FFE:FFFF:0:C800::/54
4	3FFE:FFFF:0:CC00::/54
5	3FFE:FFFF:0:D000::/54
6	3FFE:FFFF:0:D400::/54
7	3FFE:FFFF:0:D800::/54
8	3FFE:FFFF:0:DC00::/54

To create the enumerated list of subnetted network prefixes using the decimal method:

1. Based on  $s$ , the number of bits chosen for subnetting,  $m$ , the length of the network prefix being subnetted, and  $F$ , the hexadecimal value of the subnet ID being subnetted, calculate the following:

$$f = m - 48$$

$f$  is the number of bits within the prefix that are already fixed.

$$n = 2^s$$

$n$  is the number of network prefixes that are obtained.

$$i = 2^{16-(f+s)}$$

$i$  is the incremental value between each successive subnet ID.

$$l = 48+f+s$$

$l$  is the length of the new subnetted network prefixes.

$$D = \text{decimal representation of } F$$

2. Create a three-column table with  $n$  entries. The first column is the network prefix number (starting with 1), the second column is the decimal representation of the subnet ID portion of the new subnetted network prefix, and the third column is the new subnetted network prefix.
3. In the first column of the first row, the decimal representation of the subnet ID is  $D$  and the subnetted prefix is  $[48\text{-bit prefix}]:F::/l$ .
4. In the second column of the first row, increase the value of the decimal representation of the subnet ID by  $i$ . For example, in the second table entry, the decimal representation of the subnet ID is  $D+i$ .

## IPv6 Technical Reference (Microsoft Corporation)

5. In the third column of the first row, convert the decimal representation of the subnet ID to hexadecimal, and construct the prefix from [48-bit prefix]:[subnet ID]::/l. For example, in the second table entry, the subnetted network prefix is [48-bit prefix]:[D+i (converted to hexadecimal)]::/l.
6. Repeat steps 4 and 5 until the table is complete.

For example, to perform a 3-bit subnetting of the site-local network prefix 3FFE:FFFF:0:C000::/51, first calculate the values of the number of prefixes, the increment, the new prefix length, and the decimal representation of the starting subnet ID. The starting values are  $F=0xC000$ ,  $s=3$ , and  $f=51-48=3$ . The number of prefixes is 8 ( $n=2^3$ ). The increment is 1024 ( $i=2^{16-(3+3)}$ ). The new prefix length is 54 ( $l=48+3+3$ ). The decimal representation of the starting subnet ID is 49152 ( $D=0xC000=49152$ ). Next, construct a table with 8 entries. The entry for the network prefix 1 is 49192 and 3FFE:FFFF:0:C000::/54. Additional entries in the table are successive increments of  $i$  in the subnet ID portion of the network prefix, as shown in the following table.

**Decimal Subnetting for Network Prefix 3FFE:FFFF:0:C000::/51**

Network Prefix	Decimal Representation of Subnet ID	Subnetted Network Prefix
1	49192	3FFE:FFFF:0:C000::/54
2	50176	3FFE:FFFF:0:C400::/54
3	51200	3FFE:FFFF:0:C800::/54
4	52224	3FFE:FFFF:0:CC00::/54
5	53248	3FFE:FFFF:0:D000::/54
6	54272	3FFE:FFFF:0:D400::/54
7	55296	3FFE:FFFF:0:D800::/54
8	56320	3FFE:FFFF:0:DC00::/54

**Note:** RFC 3513 allows the use of subnetted network prefixes where the bits being used for subnetting are set to all 0s (the all-zeros subnetted network prefix) and all 1s (the all-ones subnetted network prefix).

### IPv6 and Name Resolution

Resolving names to IPv6 addresses is supported if either the local Hosts file (stored in `systemroot\System32\Drivers\Etc`) contains entries for IPv6 addresses or the Domain Name System (DNS) infrastructure contains resource records for IPv6 addresses. RFC 1886 describes enhancements to DNS for IPv6, and the DNS name resolver and DNS Server service in Windows Server 2003 support them.

**Note:** NetBIOS over IPv6 is not supported. Therefore, NetBIOS name resolution techniques cannot be used to resolve names to IPv6 addresses.

In RFC 1886, a new DNS resource record type, AAAA (called quad A), is used for resolving a fully qualified domain name to an IPv6 address. It is comparable to the host address (A) resource record used with IPv4. The resource record type is named AAAA (Type value of 28) because IPv6 addresses are four times as large as IPv4 addresses. The following is an example of a AAAA resource record:

## IPv6 Technical Reference (Microsoft Corporation)

`host1.microsoft.com IN AAAA FEC0::2AA:FF:FE3F:2A1C`

A host must specify either a AAAA query or a general query for a specific host name to receive IPv6 address resolution data in the DNS query answer sections.

RFC 1886 also describes the IP6.INT domain created for IPv6 reverse queries. However, according to RFC 3152, Internet Engineering Task Force (IETF) consensus has been reached that the IP6.ARPA domain be used instead of IP6.INT. Also called pointer queries, reverse queries determine a host name based on the IP address. To create the namespace for reverse queries, each hexadecimal digit in the fully expressed 32-digit IPv6 address becomes a separate level in inverse order in the reverse domain hierarchy.

For example, the reverse lookup domain name for the address FEC0::2AA:FF:FE3F:2A1C (fully expressed as FEC0:0000:0000:0000:02AA:00FF:FE3F:2A1C) is:

`C.1.A.2.F.3.E.F.F.F.0.0.A.A.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.C.E.F.IP6.ARPA.`

The DNS support that RFC 1886 describes represents a simple way to both map host names to IPv6 addresses and to provide reverse name resolution.

### IPv4 Addresses and IPv6 Equivalents

The following table lists IPv4 addresses and concepts next to their IPv6 equivalents.

**IPv4 Addresses and their IPv6 Equivalents**

IPv4 Address	IPv6 Address
Internet address classes	Not applicable in IPv6
Multicast addresses (224.0.0.0/4)	IPv6 multicast addresses (FF00::/8)
Broadcast addresses	Not applicable in IPv6
Unspecified address is 0.0.0.0	Unspecified address is ::
Loopback address is 127.0.0.1	Loopback address is ::1
Public IP addresses	Global unicast addresses
Private IP addresses (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16)	Site-local addresses (FEC0::/10)
Autoconfigured addresses (169.254.0.0/16)	Link-local addresses (FE80::/64)
Text representation: Dotted decimal notation	Text representation: Colon-hexadecimal format with suppression of leading zeros and zero compression. IPv4-compatible addresses are expressed in dotted decimal notation.
Network bits representation: Subnet mask in dotted decimal notation or prefix length notation	Network bits representation: Prefix length notation only
DNS name resolution: IPv4 host address (A) resource record	DNS name resolution: IPv6 host address AAAA resource records (RFC 1886) or A6 records (RFC 2874)

# IPv6 Technical Reference

(Microsoft Corporation)

DNS reverse resolution: IN-ADDR.ARPA domain	DNS reverse resolution: IP6.INT domain (RFC 1886) or IP6.ARPA domain (RFC 2874)
---	---

## Neighbor Discovery

Neighbor Discovery uses ICMPv6 messages to manage neighboring node interaction. Neighbor Discovery replaces ARP, ICMP Router Discovery, and ICMP Redirect and provides additional functionality.

## Neighbor Discovery Messages

All functions of Neighbor Discovery are performed with the following messages:

- **Router Solicitation:** IPv6 hosts send Router Solicitation messages to discover IPv6 routers present on the link. To prompt IPv6 routers to respond immediately, hosts send multicast Router Solicitation messages rather than waiting for a periodic Router Advertisement message.
- **Router Advertisement:** IPv6 routers send Router Advertisement messages either periodically or in response to the receipt of Router Solicitation messages. Router Advertisement messages contain the information required by hosts to determine what the link prefixes are, what the link MTU is, whether or not to use address autoconfiguration, and the duration for which addresses created through address autoconfiguration are both valid and preferred.
- **Neighbor Solicitation:** IPv6 hosts send Neighbor Solicitation messages to discover the link-layer addresses of on-link IPv6 nodes. Neighbor Solicitation messages include the link-layer address of the sender. Hosts send Neighbor Solicitation messages to multicast addresses to resolve addresses and to unicast addresses to verify the reachability of a neighboring node.
- **Neighbor Advertisement:** IPv6 nodes send Neighbor Advertisement messages in response to the receipt of Neighbor Solicitation messages. Nodes also send unsolicited Neighbor Advertisements to inform neighboring nodes of changes in link-layer addresses. Neighbor Advertisement messages contain information that nodes require to determine the type of Neighbor Advertisement message, the link-layer address of the sender, and the sender's role on the network.
- **Redirect:** IPv6 routers send Redirect messages to inform an originating host of a better first-hop address for a specific destination. Routers send Redirect messages for unicast traffic only and only to originating hosts. Only hosts process Redirect messages.

To ensure that all Neighbor Discovery messages were sent by a node on the local link, all Neighbor Discovery messages are sent with a hop limit of 255. When a Neighbor Discovery message is received, the Hop Limit field in the IPv6 header is checked. If the field is not set to 255, the message is silently discarded. Verifying that the Neighbor Discovery message has a hop limit of 255 provides protection from network attacks that are based on Neighbor Discovery and launched from off-link nodes. If a message has a hop limit of 255, a router could not have forwarded the message from an off-link node.

## Neighbor Discovery Options

RFC 2461 defines the following Neighbor Discovery options:

- **Source Link-Layer Address Option:** The Source Link-Layer Address option indicates the link-layer address of the Neighbor Discovery message sender. Neighbor Solicitation, Router

## IPv6 Technical Reference

(Microsoft Corporation)

Solicitation, and Router Advertisement messages include this option. This option is not included when the source address of the Neighbor Discovery message is the unspecified address (::).

- **Target Link-Layer Address Option:** The Target Link-Layer Address option indicates the link-layer address of the neighboring node to which IPv6 packets should be directed. Neighbor Advertisement and Redirect messages include this option.
- **Prefix Information Option:** The Prefix Information option is sent in Router Advertisement messages to indicate both address prefixes and information about address autoconfiguration. Each Router Advertisement message can include multiple Prefix Information options, indicating multiple address prefixes.
- **Redirected Header Option:** The Redirected Header option is sent in Redirect messages to specify the IPv6 packet that caused the router to send a Redirect message. This option can contain all or part of the redirected IPv6 packet, depending on the size of the IPv6 packet that was initially sent.
- **MTU Option:** The MTU option is sent in Router Advertisement messages to indicate the IPv6 MTU of the link. This option can be used when the IPv6 MTU for a link is not well known or must reflect a translational or mixed-media bridging configuration. The MTU option overrides the IPv6 MTU that the interface hardware reports.

In bridged or Layer-2 switched environments, different link-layer technologies with different link-layer MTUs can exist on the same network segment. In this case, differences in IPv6 MTUs between nodes on the same network are not discovered through Path MTU Discovery. The MTU option indicates the highest IPv6 MTU supported by all link-layer technologies on the network segment.

### Host Data Structures

To facilitate interactions between neighboring nodes, RFC 2461 defines the following host data structures as examples of how to store information for Neighbor Discovery processes:

- **Neighbor Cache:** Stores the on-link address of a neighbor, its corresponding link-layer address, and an indication of the neighbor's reachability state. The neighbor cache is equivalent to the ARP cache in IPv4.
- **Destination Cache:** Stores information about forwarding or next-hop addresses for destinations to which traffic has recently been sent. Entries in the destination cache contain the destination IP address (either local or remote), the previously resolved next-hop address, and the Path MTU for the destination.
- **Prefix List:** Lists on-link prefixes. Each entry defines a range of IP addresses for destinations that are directly reachable (neighbors). This list is populated from prefixes that routers advertise in Router Advertisement messages.
- **Default Router List:** Lists IP addresses corresponding to on-link routers that send Router Advertisement messages and that are eligible to be default routers.

RFC 2461 defines these structures as an example of an IPv6 host conceptual model. An IPv6 implementation is not required to create these exact data structures as long as the external behavior of the host is consistent with RFC 2461. For example, the IPv6 protocol for Windows Server 2003 uses a routing table rather than a prefix list and default router list. The routing table contains sufficient

## IPv6 Technical Reference (Microsoft Corporation)

information to determine how to forward IPv6 packets. For more information, see “IPv6 Routing” later in this section.

### Neighbor Discovery Message Exchanges

The Neighbor Discovery protocol provides message exchanges for the following processes:

- Address resolution
- Duplicate address detection
- Router discovery
- Redirect function

### Address Resolution

To resolve addresses, IPv6 nodes exchange Neighbor Solicitation and Neighbor Advertisement messages to resolve the link-layer address of the on-link next-hop address for a given destination. The sending host sends a multicast Neighbor Solicitation message on the appropriate interface. The multicast address of the Neighbor Solicitation message is the solicited-node multicast address derived from the target IP address. The Neighbor Solicitation message includes the link-layer address of the sending host in the Source Link-Layer Address option.

When the target host receives the Neighbor Solicitation message, it updates its own neighbor cache based on the source address of the Neighbor Solicitation message and the link-layer address in the Source Link-Layer Address option. Next, the target host sends a unicast Neighbor Advertisement to the Neighbor Solicitation sender. The Neighbor Advertisement includes the Target Link-Layer Address option.

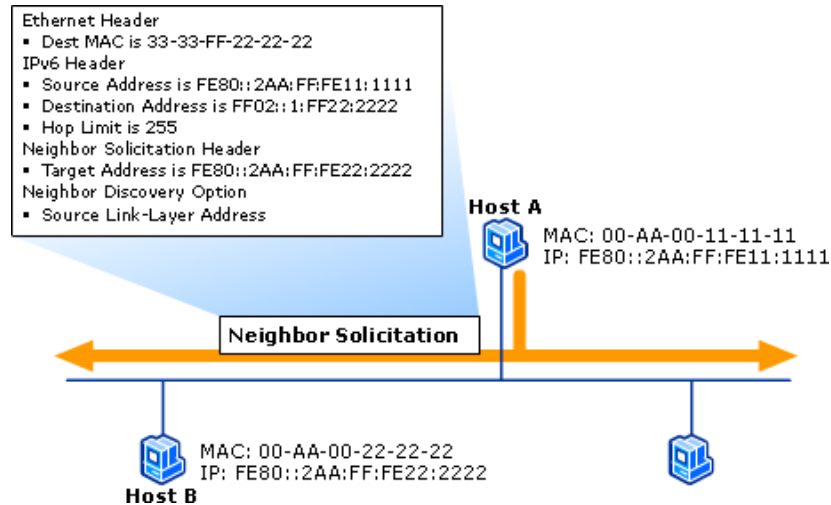
After receiving the Neighbor Advertisement from the target host, the sending host updates its neighbor cache with an entry for the target host based on the information in the Target Link-Layer Address option. At this point, the sending host and the target host can send unicast IPv6 traffic to each other.

As an example of this process, Host A has an Ethernet MAC address of 00-AA-00-11-11-11 and a corresponding link-local address of FE80::2AA:FF:FE11:1111. Host B has an Ethernet MAC address of 00-AA-00-22-22-22 and a corresponding link-local address of FE80::2AA:FF:FE22:2222. To send a packet to Host B, Host A must resolve Host B’s link-layer address.

The following figure shows Host A sending a solicited-node multicast Neighbor Solicitation message (based on Host B’s IP address) to the address of FF02::1:FF22:2222.

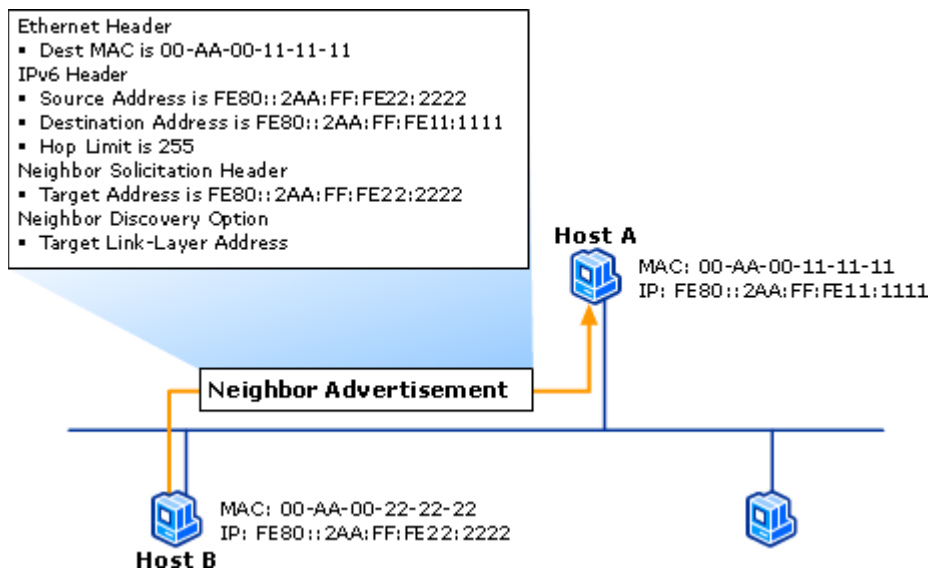
### Multicast Neighbor Solicitation Message for Address Resolution

## IPv6 Technical Reference (Microsoft Corporation)



Host B, having registered the solicited-node multicast address of 33-33-FF-22-22-22 with its Ethernet adapter, receives and processes the Neighbor Solicitation. The following figure shows Host B responding with a unicast Neighbor Advertisement message.

### Unicast Neighbor Advertisement Message for Address Resolution



### Duplicate Address Detection

IPv4 nodes use ARP Request messages and a method called gratuitous ARP to detect duplicate IP addresses on their local links. Similarly, IPv6 nodes use the Neighbor Solicitation message to detect duplicate address use on their local links.

With gratuitous ARP, the Source Protocol Address and Target Protocol Address fields in the ARP Request message header are set to the IPv4 address for which duplication is being detected. In IPv6 duplicate address detection, the Target Address field in the Neighbor Solicitation message is set to the IPv6 address for which duplication is being detected.

Duplicate address detection differs from address resolution in these ways:

## IPv6 Technical Reference

(Microsoft Corporation)

- In a Neighbor Solicitation message for duplicate address detection, the Source Address field in the IPv6 header is set to the unspecified address (::). The address being queried for duplication cannot be used until it is determined that the address is not already in use.
- In the Neighbor Advertisement reply to a Neighbor Solicitation message for duplicate address detection, the Destination Address in the IPv6 header is set to the link-local scope all-nodes multicast address (FF02::1). Because the sender of the Neighbor Solicitation message for duplicate address detection is not using the desired address, it cannot receive unicast Neighbor Advertisement messages. Therefore, the Neighbor Advertisement message is multicast.

Upon receipt of the multicast Neighbor Advertisement message sent in response to the Neighbor Solicitation message, the node disables the use of the duplicate address on the interface. If the node does not receive a Neighbor Advertisement message that defends the use of the IPv6 address, it initializes the address on the interface.

### Router Discovery

Router discovery is the process through which nodes attempt to discover the set of routers on the local link. Router discovery in IPv6 is similar to ICMP Router Discovery for IPv4 described in RFC 1256.

An important difference between ICMPv4 Router Discovery and IPv6 router discovery is the mechanism through which the default router is changed when the current one becomes unavailable. In ICMPv4 Router Discovery, the Router Advertisement message includes an Advertisement Lifetime field. Advertisement Lifetime is the time after which the router, upon sending its last Router Advertisement message, can be considered unavailable. In the worst case, a router can become unavailable and hosts will not attempt to discover a new default router until the Advertisement Lifetime has elapsed.

IPv6 has a Router Lifetime field in the Router Advertisement message. This field indicates the length of time that the router can be considered a default router. However, if the current default router becomes unavailable, the condition is detected through neighbor unreachability detection instead of the Router Lifetime field in the Router Advertisement message. Because neighbor unreachability detection determines that the router is no longer reachable, a new router is chosen immediately from the default router list. For more information, see "Neighbor Unreachability Detection" later in this section.

In addition to configuring a default router, IPv6 router discovery also configures the following:

- The default setting for the Hop Limit field in the IPv6 header.
- A determination of whether the node should use a stateful address protocol, such as Dynamic Host Configuration Protocol for IPv6 (DHCPv6), for addresses and other configuration parameters.
- The timers used in reachability detection and the retransmission of Neighbor Solicitation messages.
- The list of network prefixes defined for the link. Each network prefix contains both the IPv6 network prefix and its valid and preferred lifetimes. If indicated, a network prefix combined with the interface identifier creates a stateless IP address configuration for the receiving interface. A network prefix also defines the range of addresses for nodes on the local link.

## IPv6 Technical Reference (Microsoft Corporation)

- The MTU of the local link.

The IPv6 router discovery processes are the following:

- IPv6 routers periodically send Router Advertisement messages on the local link advertising their existence as routers. They also provide configuration parameters such as default hop limit, MTU, and prefixes.
- Active IPv6 hosts on the local link receive the Router Advertisement messages and use the contents to maintain their default router lists, prefix lists, and other configuration parameters.
- A host that is starting up sends a Router Solicitation message to the link-local scope all-routers multicast address (FF02::2). Upon receipt of a Router Solicitation message, each router on the local link send a unicast Router Advertisement message to the node that sent the Router Solicitation message. The node receives the Router Advertisement messages and uses their contents to build the default router and prefix lists and to set other configuration parameters.

### Redirect Function

Routers use the redirect function to inform originating hosts of a better first-hop neighbor to which traffic should be forwarded for a specific destination. Routers use the redirect function for two purposes:

- A router informs an originating host of the IP address of a router available on the local link that is closer to the destination. The term closer is a routing metric function used to reach the destination network segment. This condition can occur when multiple routers are on a network segment, the originating host chooses a default router, and it is not the best one to use to reach the destination.
- A router informs an originating host that the destination is a neighbor (it is on the same link as the originating host). This condition can occur when the prefix list of a host does not include the prefix of the destination. Because the destination does not match a prefix in the list, the originating host forwards the packet to its default router.

The following steps occur in the IPv6 redirect process:

1. The originating host sends a unicast packet to its default router.
2. The router processes the packet and notes that the address of the originating host is a neighbor. Additionally, the router notes that both the originating host and the next-hop are on the same link.
3. The router forwards the packet to the appropriate next-hop address.
4. The router sends the originating host a Redirect message. In the Target Address field of the Redirect message is the next-hop address of the node to which the originating host should send packets addressed to the destination. For packets redirected to a router, the Target Address field is set to the link-local address of the router. For packets redirected to a host, the Target Address field is set to the destination address of the packet originally sent. The Redirect message includes the Redirected Header option. The message might also include the Target Link-Layer Address option.

## IPv6 Technical Reference

(Microsoft Corporation)

5. Upon receiving the Redirect message, the originating host updates the destination address entry in the destination cache with the address in the Target Address field. If the Redirect message includes the Target Link-Layer Address option, its contents are used to create or update the corresponding entry in the neighbor cache.

Only the first router in the path between the originating host and the destination sends redirect messages, and (like ICMPv6 error messages) they are rate limited. Hosts never send Redirect messages, and routers never update routing tables based on the receipt of a Redirect message.

### Neighbor Unreachability Detection

Reachability is defined as the ability to send an IPv6 packet to a neighboring node and have the IPv6 layer of the neighbor receive and process that packet. When a node sends a packet to a router, the packet is delivered to the router's IPv6 layer and then forwarded to the next hop. When a node sends a packet to a neighboring node, the packet is delivered to the node's IPv6 layer. It is important to note that the definition of reachability does not require delivery to a remote node across a router — only to the neighboring router.

When a neighbor becomes unreachable, IPv6 detects this condition and attempts to correct it. To determine whether a neighbor is reachable, IPv6 relies on either upper layer protocols that indicate communication progress or receipt of a Neighbor Advertisement message that has been sent in response to a unicast Neighbor Solicitation message.

For TCP traffic, communication progress is indicated when new data or acknowledgement segments for sent data are received. For UDP traffic, a progress indication might not be present. In this case, the node sends unicast Neighbor Solicitation messages to the next-hop neighbor to monitor its ongoing reachability.

Only the receipt of a solicited Neighbor Advertisement message is considered proof of reachability. A solicited Neighbor Advertisement message, which has its Solicited flag set to 1, is sent only in response to a Neighbor Solicitation message. Unsolicited Neighbor Advertisement or Router Advertisement messages are not considered proof of reachability.

Neighbor unreachability detection detects symmetric reachability. In this instance, packets must be able to travel to and from the desired neighboring node. When a Neighbor Solicitation message is sent and a solicited Neighbor Advertisement message is received, the path between the nodes is confirmed. For an unsolicited Neighbor Advertisement or Router Advertisement message, only the path from the node sending the message is confirmed. This is called asymmetric reachability.

For a specific local node, reachability is confirmed only by the node that sends the Neighbor Solicitation message and receives the Neighbor Advertisement message. The node sending the Neighbor Advertisement message receives no confirmation that the message reached the intended node. For two neighboring nodes to both determine reachability, each node must exchange Neighbor Solicitation and Neighbor Advertisement messages with the other.

The reachability of a neighboring node is determined by monitoring the state of the neighboring node's entry in the neighbor cache. RFC 2461 defines the following states for a neighbor cache entry:

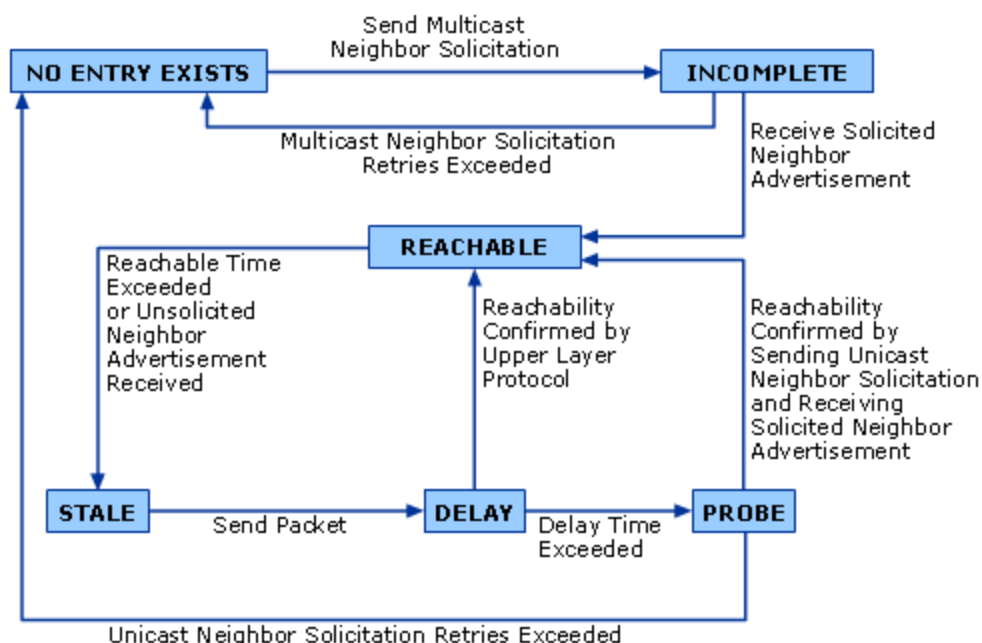
- **INCOMPLETE:** IPv6 address resolution, which is using a solicited-node multicast Neighbor Solicitation message, is in progress. The INCOMPLETE state is entered when a neighbor cache entry is created but does not yet have the node's corresponding link-layer address. RFC 2461 recommends that three successive multicast Neighbor Solicitation messages be sent.

## IPv6 Technical Reference (Microsoft Corporation)

- **REACHABLE:** Reachability has been confirmed by receipt of a solicited unicast Neighbor Advertisement message. The neighbor cache entry stays in the REACHABLE state until the number of milliseconds indicated in the Reachable Time field in the Router Advertisement message elapses. The entry stays in the REACHABLE state as long as upper layer protocols such as TCP indicate that communication is progressing. Each time that progress is indicated, the reachable time for the entry is refreshed.
- **STALE:** Reachable time (the duration since the last reachability confirmation was received) has elapsed. The neighbor cache entry enters the STALE state after the number of milliseconds in the Reachable Time field in the Router Advertisement message (or a host default value) elapses, and the entry remains in this state until a packet is sent to the neighbor. The entry also enters the STALE state when the host receives an unsolicited Neighbor Advertisement message that is advertising the link-layer address.
- **DELAY:** To allow time for upper layer protocols to provide reachability confirmation before sending Neighbor Solicitation messages, the neighbor cache entry enters the DELAY state and waits a configurable period of time after sending a packet. RFC 2461 recommends a value of 5 seconds. If reachability is not confirmed by the delay time, then the entry enters the PROBE state, and a unicast Neighbor Solicitation message is sent.
- **PROBE:** Reachability confirmation is in progress for a neighbor cache entry that was in the STALE or DELAY state. Unicast Neighbor Solicitation messages are sent at intervals corresponding to a retransmission timer field in the Router Advertisement message that this host received. A configurable variable determines the number of Neighbor Solicitation messages sent before the reachability detection process is abandoned and the neighbor cache entry is removed. RFC 2461 recommends sending three successive unicast Neighbor Solicitation messages.

The following figure shows the state diagram of an entry in the neighbor cache.

**States of a Neighbor Cache Entry**



# IPv6 Technical Reference

(Microsoft Corporation)

If the unreachable neighbor is a router, the host chooses another router from the default router list and performs both address resolution and unreachability detection on it.

## Address Autoconfiguration

A highly useful aspect of IPv6 is its ability to automatically configure itself without the use of a stateful configuration protocol, such as Dynamic Host Configuration Protocol for IPv6 (DHCPv6). By default, an IPv6 host can configure a link-local address for each interface. By using router discovery, a host can also determine the addresses of routers, additional addresses, and other configuration parameters. The Router Advertisement message indicates whether a stateful address configuration protocol should be used.

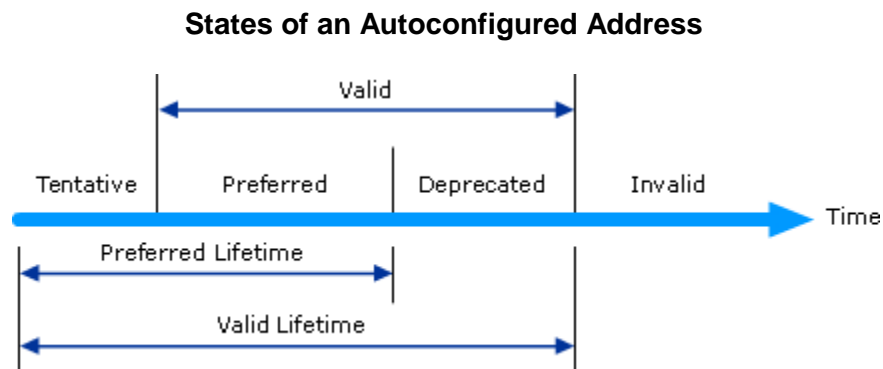
Address autoconfiguration can be performed only on multicast-capable interfaces. RFC 2462 describes address autoconfiguration.

## Autoconfigured Address States

Autoconfigured addresses are in one or more of the following states:

- **Tentative:** The address is in the process of being verified as unique. Verification occurs through duplicate address detection.
- **Preferred:** An address for which uniqueness has been verified. A node can send and receive unicast traffic to and from a preferred address. Router Advertisement messages include the period of time that an address can remain in the tentative and preferred states.
- **Deprecated:** An address that is still valid but whose use is discouraged for new communication. Existing communication sessions can continue to use a deprecated address. Nodes can send and receive unicast traffic to and from deprecated addresses.
- **Valid:** An address from which unicast traffic can be sent and received. The valid state covers both the preferred and deprecated states. Router Advertisement messages include the amount of time that an address remains in the valid state. The valid lifetime must be longer than or equal to the preferred lifetime.
- **Invalid:** An address for which a node can no longer send or receive unicast traffic. An address enters the invalid state after the valid lifetime expires.

The following figure shows the relationship between the states of an autoconfigured address and the preferred and valid lifetimes.



# IPv6 Technical Reference

## (Microsoft Corporation)

With the exception of link-local addresses, address autoconfiguration is specified only for hosts. Routers must obtain address and configuration parameters through another means (for example, manual configuration).

### Types of Autoconfiguration

Autoconfiguration falls into three types:

1. **Stateless:** Configuration is based on Router Advertisement messages. These messages include stateless address prefixes and require that hosts not use a stateful address configuration protocol.
2. **Stateful:** Configuration is based on a stateful address configuration protocol, such as DHCPv6, to obtain addresses and other configuration options. Hosts use stateful address configuration when they receive Router Advertisement messages that do not include address prefixes and that require the hosts to use a stateful address configuration protocol. A host will also use a stateful address configuration protocol when no routers are present on the local link.
3. **Both:** Configuration is based on Router Advertisement messages. These messages include stateless address prefixes but require hosts to use a stateful address configuration protocol.

For all autoconfiguration types, a link-local address is always configured.

### Autoconfiguration Process

Address autoconfiguration for an IPv6 node occurs as follows:

1. A tentative link-local address is derived, based on the link-local prefix of FE80::/64 and the 64-bit interface identifier.
2. Duplicate address detection is performed to verify the uniqueness of the tentative link-local address. If the address is already in use, the node must be configured manually. If the address is not already in use, the tentative link-local address is assumed to be unique and valid. The link-local address is initialized for the interface. The corresponding solicited-node multicast link-layer address is registered with the network adapter.
3. The host sends a Router Solicitation message.
4. If the host receives no Router Advertisement messages, then it uses a stateful address configuration protocol to obtain addresses and other configuration parameters. Windows Server 2003 does not support the use of a stateful address configuration protocol for IPv6.
5. If the host receives a Router Advertisement message, the host is configured based on the information in the message
6. For each stateless address prefix that the message includes: A tentative address is derived from the address prefix and the appropriate 64-bit interface identifier. The uniqueness of the tentative address is verified. If the tentative address is in use, the address is not initialized for the interface. If the tentative address is not in use, the address is initialized. Initialization includes setting the valid and preferred lifetimes based on information in the Router Advertisement message. Initialization also includes registering the corresponding solicited-node multicast link-layer address with the network adapter.

## IPv6 Technical Reference (Microsoft Corporation)

7. If specified in the Router Advertisement message, the host uses a stateful address configuration protocol to obtain additional addresses or configuration parameters.

### IPv6 Routing

Similar to IPv4, IPv6 nodes use local routing tables to determine how to forward packets. IPv6 routing table entries are created by default when IPv6 initializes, and either a system administrator adds entries manually or communication with routers adds them automatically.

A routing table is present on each IPv6 node that is running Windows Server 2003. The routing table stores information about IPv6 network prefixes and how they can be reached (either directly or indirectly). Routing tables are not exclusive to IPv6 routers. IPv6 hosts that are running Windows Server 2003 also have IPv6 routing tables. Before a routing table is used, the destination cache is checked for an entry matching the destination address in the packet being forwarded. If the destination cache does not contain an entry for the destination address, the routing table is used to determine:

1. The ext-hop Address: For a direct delivery (in which the destination is on a local link), the next-hop address is the destination address in the packet. For an indirect delivery (in which the destination is not on a local link), the next-hop address is the address of a router.
2. The Next-hop Interface: The interface identifies the physical or logical interface that is used to forward the packet either to its destination or to the next router.

After the next-hop address and interface are determined, the destination cache is updated. Subsequent packets forwarded to the destination use the destination cache entry, rather than the routing table.

### Entry Types in IPv6 Routing Tables

Entries in IPv6 routing tables store the following types of routes:

- **Directly Attached Network Routes:** Routes for network prefixes that are directly attached and typically have prefixes that are 64 bits long.
- **Remote Network Routes:** Routes for network prefixes that are not directly attached but are available across other routers. Remote network routes can be subnet network prefixes or prefixes for address spaces.
- **Host Routes:** Routes to specific IPv6 addresses. Host routes allow routing to occur for each IPv6 address. For host routes, the route prefix is a specific IPv6 address with a prefix length of 128 bits. Prefixes for both types of network routes are shorter than 128 bits.
- **Default Route:** The default route is used when a more specific network route or host route is not found. The default route prefix is `::/0`.

### Route Determination Process

IPv6 uses the following process to determine which routing table entry is used for the forwarding decision:

- For each entry in a routing table, IPv6 compares the bits in the network prefix to the same bits in the destination IPv6 address for the number of bits in the prefix length in the route. If all the bits in

## IPv6 Technical Reference (Microsoft Corporation)

the network prefix match all the bits in the destination IPv6 address for the number of bits in the prefix length for the route, the route is a match for the destination.

- The list of matching routes is compiled. The route that has the largest prefix length (the route that matched the most high-order bits with the destination IPv6 address) is chosen. The longest matching route is the most specific route to the destination. If multiple entries with the longest match are found (multiple routes to the same network prefix, for example), the router uses the lowest metric to identify the best route. If multiple entries exist that are the longest match and the lowest metric, IPv6 can choose which routing table entry to use.

The match for any given destination finds routes in the following order:

1. A route that matches the entire destination IPv6 address (a host route with a 128-bit prefix length).
2. A route with the longest prefix length that matches the destination.
3. The default route (the network prefix `::/0`).

The end result of the route determination process is the selection of a single route in the routing table. The selected route yields a next-hop address and an interface. If the route determination process on the sending host fails to find a route, IPv6 assumes that the destination is locally reachable. If the route determination process on a router fails to find a route, IPv6 sends an ICMPv6 Destination Unreachable-No Route Found message to the sending host and discards the packet.

### End-to-End IPv6 Delivery Process

The following sections describe the process of forwarding an IPv6 packet from the sending host across one or more IPv6 routers to the final destination. These sections assume that the Hop-by-Hop Options, Destination Options, and Routing extension headers are not present.

### IPv6 on the Sending Host

The process by which hosts send IPv6 packets combines the local host's data structures with the Neighbor Discovery protocol. IPv6 hosts use the following algorithm when sending a packet to an arbitrary destination:

1. Set the Hop Limit value to either a default or application-specified value.
2. Check the destination cache for an entry that matches the destination address.
3. If the destination cache contains an entry that matches the destination address, obtain the next-hop address and interface index from the destination cache entry. If the destination is a mobile IPv6 node and the sending host is a mobile correspondent node, the destination cache entry might contain a pointer to care-of destination cache. If so, obtain the next-hop address from the corresponding entry in the care-of destination cache. For more information about IPv6 mobility support, see the Internet draft titled "Mobility Support in IPv6." Go to step 7.
4. If the destination cache does not contain an entry that matches the destination address, check the local IPv6 routing table for the longest matching route to the destination address.
5. Based on the longest matching route, determine the next-hop address and the interface index used for forwarding the packet. If no route is found, IPv6 assumes that the destination is directly reachable. The next-hop address is set to the destination address, and an interface index is chosen.

## IPv6 Technical Reference (Microsoft Corporation)

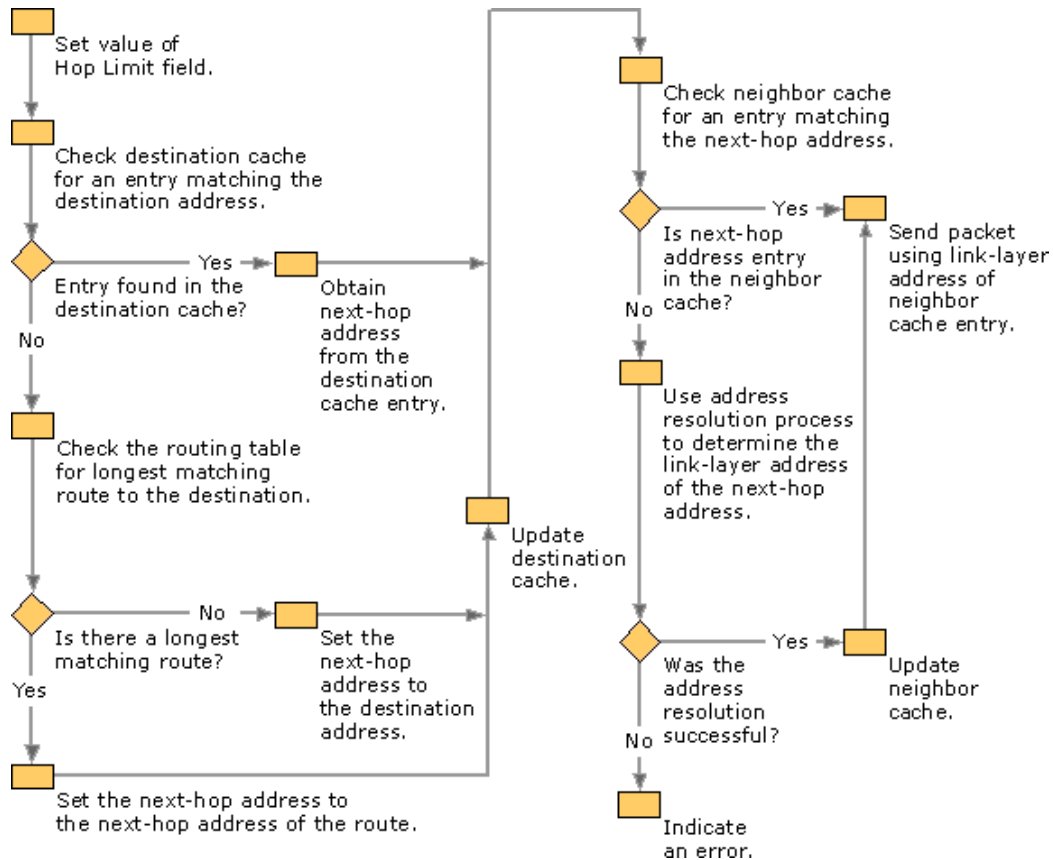
6. Update the destination cache.
7. Check the neighbor cache for an entry that matches the next-hop address.
8. If the neighbor cache contains an entry that matches the next-hop address, obtain the link-layer address.
9. If the neighbor cache does not contain an entry that matches the next-hop address, use address resolution to obtain the link-layer address for the next-hop address. If address resolution is not successful, indicate an error.
10. Send the packet using the link-layer address of the entry in the neighbor cache.

The following figure shows this sending host process.

# IPv6 Technical Reference

(Microsoft Corporation)

## Sending Host Process



## IPv6 on the Router

IPv6 routers use the following algorithm when they receive and forward a packet to an arbitrary destination:

1. Verify whether the destination address in the IPv6 packet corresponds to an address assigned to a router interface. If so, the router processes the IPv6 packet as the destination host. (For more information, see step 2 in the following subsection, "IPv6 on the Destination Host.")
2. Decrement the value of the Hop Limit field by 1. If the value of the Hop Limit field is 0, send an ICMPv6 Time Exceeded message to the sender, and discard the packet.
3. If the value of the Hop Limit field is 1 or greater, update the Hop Limit field in the IPv6 header of the packet.
4. Check the destination cache for an entry that matches the destination address.
5. If the destination cache contains an entry that matches the destination address, obtain the next-hop address and interface index in the entry, and go to step 9.
6. Check the local IPv6 routing table for the longest matching route to the destination address.

## IPv6 Technical Reference

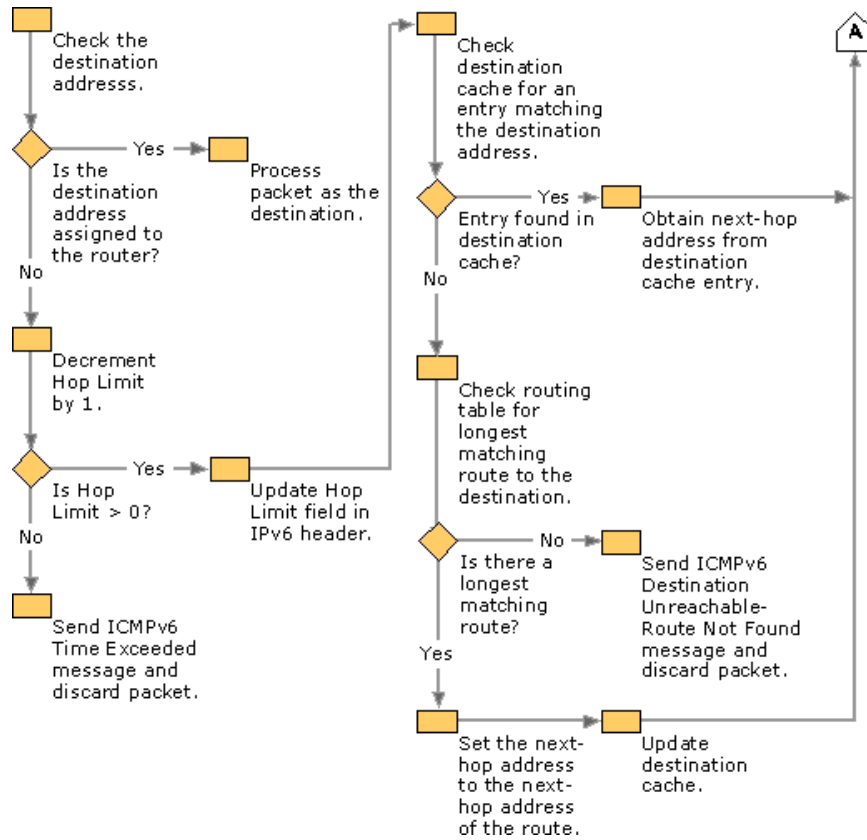
(Microsoft Corporation)

7. Based on the longest matching route, determine the next-hop IPv6 address and the interface index used for forwarding the packet. If no route is found, send an ICMPv6 Destination Unreachable-No Route Found message to the sending host, and discard the packet.
8. Update the destination cache.
9. If the interface on which the packet was received is the same as the interface on which the packet is being forwarded, the interface is a point-to-point link, and the Destination Address field matches a prefix assigned to the interface, send an ICMPv6 Destination Unreachable-Address Unreachable message to the sending host, and discard the packet. This prevents the needless “ping-pong” forwarding of IPv6 packets between the two interfaces on a point-to-point link for a packet whose destination matches the prefix of the point-to-point link but does not match the address of either interface. This condition and its solution is described in the Internet draft titled “Avoiding ping-pong packets on point-to-point links.”
10. If the interface on which the packet was received is the same as the interface on which the packet is being forwarded and the Source Address field matches a prefix assigned to the interface, send a Redirect message to the sending host (subject to rate limiting).
11. Compare the link MTU of the next-hop interface to the size of the IPv6 packet being forwarded.
12. If the link MTU is smaller than the packet, send an ICMPv6 Packet Too Big message.
13. Check the neighbor cache of the next-hop interface for an entry that matches the next-hop address.
14. If the neighbor cache of the next-hop interface contains an entry that matches the next-hop address, obtain the link-layer address.
15. If the neighbor cache of the next-hop interface does not contain an entry that matches the next-hop address, use address resolution to obtain the link-layer address for the next-hop address. If address resolution is not successful, send an ICMPv6 Destination Unreachable-Destination Address Unreachable message, and discard the packet.
16. Send the packet using the link-layer address of the neighbor cache entry.

The following figures show this router forwarding process.

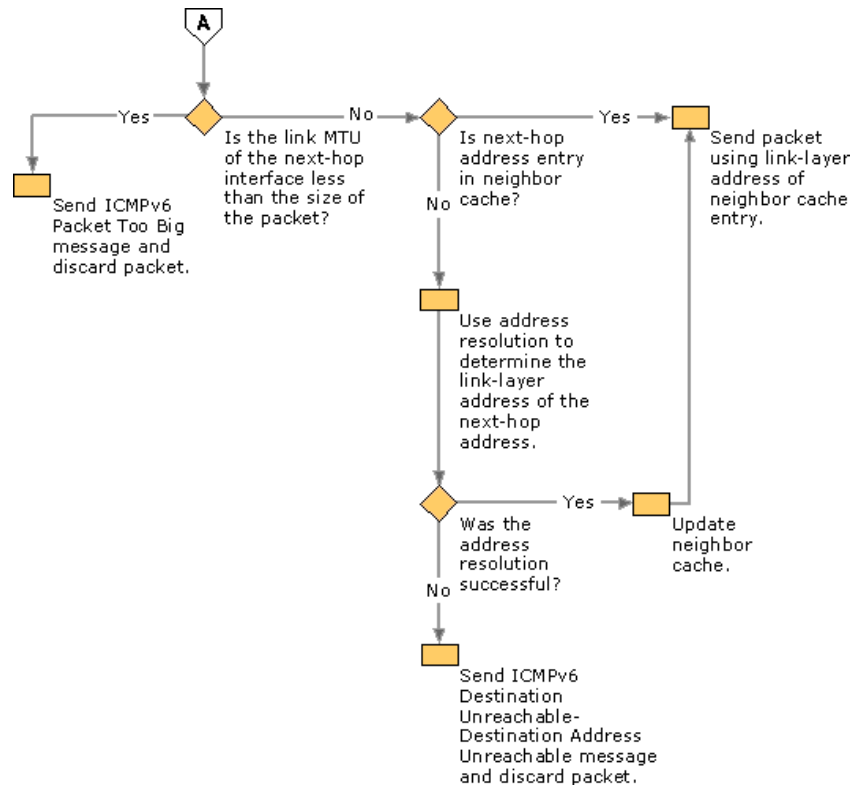
# IPv6 Technical Reference (Microsoft Corporation)

## Router Forwarding Process



## Router Forwarding Process

## IPv6 Technical Reference (Microsoft Corporation)



This entire process is repeated at each router in the path between the source and destination host.

### IPv6 on the Destination Host

IPv6 hosts use the following algorithm when they receive IPv6 packets:

1. Verify whether the destination address in the packet corresponds to an IPv6 address that is assigned to a host interface. If the destination address is not assigned to a host interface, silently discard the packet.
2. Based on the Next Header field, process extension headers (if present), and pass the upper-layer PDU to the appropriate upper-layer protocol. If the protocol does not exist, send an ICMPv6 Parameter Problem message back to the sender, and discard the packet.
3. For TCP and UDP packets, check the destination port. If no application exists for the UDP port number, send an ICMPv6 Destination Unreachable-Destination Port Unreachable message back to the sender, and discard the packet. If no application exists for the TCP port number, send a TCP Connection Reset segment back to the sender, and discard the packet.
4. For TCP and UDP packets, pass the TCP segment or UDP message to the appropriate application.

### IPv6 Internet Connection Firewall

A firewall provides security by creating a protective boundary between a computer or network and the Internet. When you install the Advanced Networking Pack for Windows XP, IPv6 Internet Connection Firewall (ICF) is provided and enabled by default to dynamically restrict traffic from the Internet. IPv6 ICF is different from the existing ICF in Windows XP, which restricts IPv4 traffic.

## IPv6 Technical Reference

(Microsoft Corporation)

### IPv6 ICF:

- Runs automatically and provides filters for all network connections on which IPv6 is enabled.
- Monitors all outbound traffic and dynamically creates inbound packet filters for response traffic. This is known as stateful filtering. IPv6 ICF silently discards all unsolicited inbound traffic.
- Logs IPv6 traffic events to a log file (separate from IPv4 ICF). By default, this log file is located at: `systemroot\Pfirewall-v6.log`.

IPv6 ICF protects your computer by silently dropping all packets that are initiated from a source outside the IPv6 ICF computer, such as the Internet. IPv6 ICF stops common attempts to illegally gain access to your computer or network by malicious Internet users using techniques such as port scanning. Instead of notifying you of inbound discarded traffic, IPv6 ICF creates entries in a security log from which you can view the activity that the firewall tracks.

IPv6 ICF is configured using commands in the netsh firewall context. You can use netsh commands to configure IPv6 ICF to allow specific types of ICMPv6 traffic or traffic to specific TCP or UDP ports. Note: Windows XP shows only the IPv4 ICF configuration in Network Connections. IPv6 ICF might appear disabled, but it is actually enabled and filtering IPv6 traffic.

# IPv6 Technical Reference

(Microsoft Corporation)

## IPv6 Transition Technologies

Protocol transitions are not easy, and the transition from IPv4 to IPv6 is no exception. Protocol transitions are typically deployed by installing and configuring the new protocol on all nodes within the network and verifying that all node and router operations work. Although this might be possible in a small- or medium-sized organization, the challenge of making a rapid protocol transition in a large organization is very difficult. Additionally, given the scope of the Internet, rapid protocol transition becomes an impossible task.

The transition from IPv4 to IPv6 will take years, and organizations or hosts within organizations might continue to use IPv4 indefinitely. Therefore, although migration is the long-term goal, equal consideration must be given to the interim coexistence of IPv4 and IPv6 nodes.

RFC 1752 defines the following transition criteria:

- Existing IPv4 hosts can be upgraded at any time, independent of the upgrade of other hosts or routers.
- Hosts that use only IPv6 can be added at any time, without dependencies on other hosts or routing infrastructure.
- IPv4 hosts on which IPv6 is installed can continue to use their IPv4 addresses and do not need additional addresses.
- Little preparation is required to either upgrade IPv4 nodes to IPv6 or deploy new IPv6 nodes.

The inherent lack of dependencies between IPv4 and IPv6 hosts, IPv4 routing infrastructure, and IPv6 routing infrastructure requires several mechanisms that allow seamless coexistence.

## Node Types

RFC 2893 defines the following node types:

- **IPv4-only Node:** A node that implements only IPv4 (and has only IPv4 addresses). This node does not support IPv6. Most hosts and routers installed today are IPv4-only nodes.
- **IPv6-only Node:** A node that implements only IPv6 (and has only IPv6 addresses). This node is able to communicate only with IPv6 nodes and applications. This type of node is not common today, but it might become more prevalent as smaller devices such as cellular phones and handheld computing devices include the IPv6 protocol.
- **IPv6/IPv4 Node:** A node that implements both IPv4 and IPv6. This node is IPv6-enabled if it has an IPv6 interface configured.
- **IPv4 Node:** A node that implements IPv4 (it can send and receive IPv4 packets). An IPv4 node can be an IPv4-only node or an IPv6/IPv4 node.
- **IPv6 Node:** A node that implements IPv6 (it can send and receive IPv6 packets). An IPv6 node can be an IPv6-only node or an IPv6/IPv4 node.

For coexistence to occur, the largest number of nodes (IPv4 or IPv6 nodes) can communicate using an IPv4 infrastructure, an IPv6 infrastructure, or an infrastructure that is a combination of IPv4 and IPv6. True migration is achieved when all IPv4 nodes are converted to IPv6-only nodes. However, for the foreseeable future, practical migration is achieved when as many IPv4-only nodes as possible are

## IPv6 Technical Reference (Microsoft Corporation)

converted to IPv6/IPv4 nodes. IPv4-only nodes can communicate with IPv6-only nodes only through an IPv4-to-IPv6 proxy or translation gateway.

### Address Compatibility

The following addresses are defined to help IPv4 and IPv6 nodes coexist:

- **IPv4-compatible Addresses:** IPv6/IPv4 nodes that are communicating with IPv6 over an IPv4 infrastructure use IPv4-compatible addresses, 0:0:0:0:0:w.x.y.z or ::w.x.y.z (where w.x.y.z is the dotted decimal representation of a public IPv4 address). When an IPv4-compatible address is used as an IPv6 destination, IPv6 traffic is automatically encapsulated with IPv4 headers and sent to their destinations using the IPv4 infrastructure.
- **IPv4-mapped Addresses:** The IPv4-mapped address, 0:0:0:0:FFFF:w.x.y.z or ::FFFF:w.x.y.z, is used to represent an IPv4-only node to an IPv6 node. It is used only for internal representation. The IPv4-mapped address is never used as a source or destination address of an IPv6 packet. The IPv6 protocol for Windows Server 2003 does not support IPv4-mapped addresses. Some IPv6 implementations use IPv4-mapped addresses when translating traffic between IPv4-only and IPv6-only nodes.
- **6over4 Addresses:** Each 6over4 address comprises a valid 64-bit unicast address prefix and the interface identifier ::WWXX:YYZZ (where WWXX:YYZZ is the colon-hexadecimal representation of w.x.y.z, a unicast IPv4 address assigned to an interface). An example of a link-local 6over4 address based on the IPv4 address of 131.107.4.92 is FE80::836B:45C. 6over4 addresses represent a host that use the automatic tunneling mechanism defined in RFC 2529.
- **6to4 Addresses:** 6to4 addresses are based on the prefix 2002:WWXX:YYZZ::/48 (where WWXX:YYZZ is the colon-hexadecimal representation of w.x.y.z, a public IPv4 address assigned to an interface). 6to4 addresses represent sites that use the automatic tunneling mechanism defined in RFC 3056, also known as 6to4. For more information, see “6to4” later in this section.
- **ISATAP Addresses:** Each Intra-site Automatic Tunnel Addressing Protocol (ISATAP) addresses comprise a valid 64-bit unicast address prefix and the interface identifier ::0:5EFE:w.x.y.z (where w.x.y.z is a unicast IPv4 address assigned to an interface). An example of a link-local ISATAP address is FE80::5EFE:131.107.4.92. ISATAP addresses represent hosts that use the automatic tunneling mechanism defined in the Internet draft titled “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP).” For more information, see “ISATAP” later in this section.
- **Teredo Addresses:** Teredo addresses use the prefix 3FFE:831F::/32. An example of a Teredo address is 3FFE:831F:CE49:7601:8000:FFFF:62C3:FFFE. Beyond the first 32 bits, Teredo addresses encode the IPv4 address of a Teredo server, flags, and the encoded version of the external address and port of a Teredo client. Teredo addresses represent hosts that use the automatic tunneling mechanism defined in the Internet draft titled “Teredo: Tunneling IPv6 over UDP through NATs.” For more information, see “Teredo” later in this section.

### Transition Mechanisms

The following mechanisms are used to help IPv6 coexist with an IPv4 infrastructure and to provide an eventual transition to an IPv6-only infrastructure:

- Dual IP layer
- IPv6 over IPv4 tunneling
- DNS infrastructure

# IPv6 Technical Reference

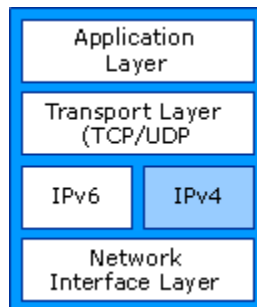
(Microsoft Corporation)

## Dual IP Layer

The dual IP layer is an implementation of the TCP/IP suite of protocols that includes both an IPv4 Internet layer and an IPv6 Internet layer. IPv6/IPv4 nodes use this mechanism to communicate with both IPv4 and IPv6 nodes. A dual IP layer contains a single implementation of Host-to-Host Transport layer protocols such as TCP and UDP. All upper layer protocols in an implementation of dual IP layer can communicate over IPv4, IPv6, or IPv6 tunneled in IPv4.

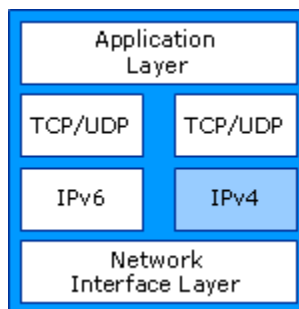
The following figure shows a dual IP layer architecture.

### A Dual IP Layer Architecture



The IPv6 protocol for Windows Server 2003 is not a dual IP layer. The IPv6 protocol driver, Tcpi6.sys, contains a separate implementation of TCP and UDP and is sometimes referred to as a dual-stack implementation. The following figure shows the dual stack architecture of the IPv6 protocol for Windows Server 2003.

### The Dual Stack Architecture of the IPv6 Protocol for Windows Server 2003



Although the IPv6 protocol for Windows Server 2003 is not a dual IP layer, it functions in the same way as a dual IP layer in terms of providing functionality for IPv6 transition.

## IPv6 over IPv4 Tunneling

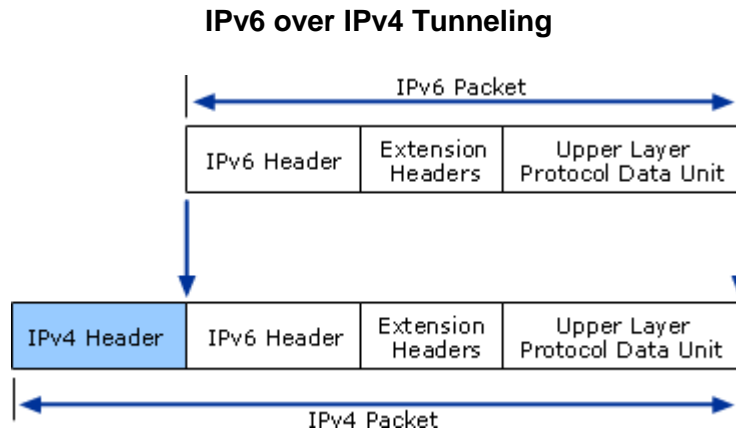
IPv6 over IPv4 tunneling is the encapsulation of IPv6 packets with an IPv4 header so that IPv6 packets can be sent over an IPv4 infrastructure. Within the IPv4 header:

- The IPv4 Protocol field is set to 41 to indicate an encapsulated IPv6 packet.
- The Source and Destination fields are set to IPv4 addresses of the tunnel endpoints. The tunnel endpoints are either manually configured as part of the tunnel interface or are automatically

## IPv6 Technical Reference (Microsoft Corporation)

derived from the sending interface, the next-hop address of the matching route, or the source and destination IPv6 addresses in the IPv6 header.

The following figure shows IPv6 over IPv4 tunneling.



For IPv6 over IPv4 tunneling, the IPv6 path MTU is typically 20 less than the IPv4 path MTU. However, if the IPv4 path MTU is not stored for each tunnel, an intermediate IPv4 router might need to fragment the IPv4 packet. In this case, the Don't Fragment flag in the IPv4 header must be set to 0 when IPv6 over IPv4 tunneled packets are sent.

### DNS Infrastructure

Because most users refer to network resources by name rather than by address, successful coexistence requires upgrading the DNS infrastructure. Upgrading the DNS infrastructure consists of populating the DNS servers with records to support IPv6 name-to-address and address-to-name resolution. After the sending host obtains addresses using a DNS name query, the node must select which addresses to use for communication.

### Address Records

The DNS infrastructure must contain the following resource records (populated either manually or dynamically) for the successful resolution of domain names to addresses:

- A records for IPv4-only and IPv6/IPv4 nodes
- AAAA records for IPv6-only and IPv6/IPv4 nodes

### Pointer Records

The DNS infrastructure must contain the following resource records (populated either manually or dynamically) for the successful resolution of address to domain names (reverse queries):

- PTR records in the IN-ADDR.ARPA domain for IPv4-only and IPv6/IPv4 nodes
- PTR records in the IP6.ARPA domain for IPv6-only and IPv6/IPv4 nodes (optional)

### Address Selection Rules

For name-to-address resolution, after the querying node obtains the set of addresses corresponding to the name, the node must determine the set of addresses to choose as source and destination for outbound packets.

## IPv6 Technical Reference (Microsoft Corporation)

This choice is not an issue in today's environment, in which IPv4-only nodes prevail. However, in an environment in which IPv4 and IPv6 coexist, the set of addresses returned in a DNS query might contain multiple IPv4 and IPv6 addresses. The querying host is configured with at least one IPv4 address and (typically) multiple IPv6 addresses. It is not an easy task to decide which type of address (IPv4 versus IPv6) and then which scope (public versus private for IPv4 and link-local versus global versus compatible for IPv6) to use for both the source and the destination addresses.

Default address selection rules are currently under discussion and are described in RFC 3484. You can view the default address selection rules for the IPv6 protocol for Windows Server 2003 using the `netsh interface ipv6 show prefixpolicy` command to display the prefix policy table. You can modify the entries in the prefix policy table using the `netsh interface ipv6 add|set|delete prefixpolicy` commands. By default, IPv6 addresses in DNS query responses are preferred over IPv4 addresses.

### Tunneling Configurations

RFC 2893 defines the following configurations with which to tunnel IPv6 traffic between IPv6/IPv4 nodes over an IPv4 infrastructure:

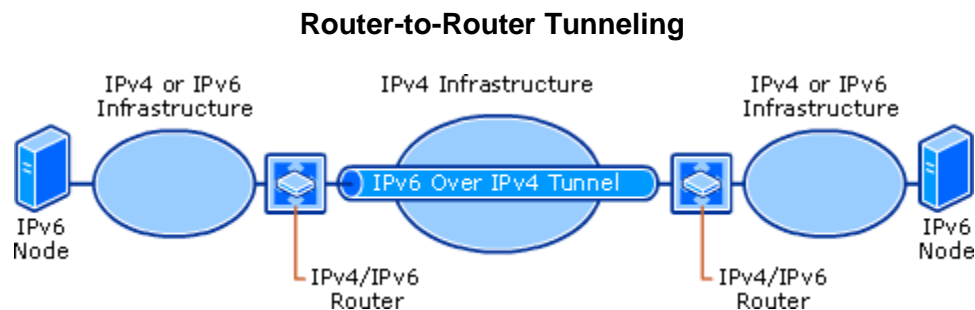
- Router-to-Router
- Host-to-Router or Router-to-Host
- Host-to-Host

Note: IPv6 over IPv4 tunneling describes only an encapsulation of IPv6 packets with an IPv4 header so that IPv6 nodes are reachable across an IPv4 infrastructure. Unlike tunneling for Point-to-Point Tunneling Protocol (PPTP) and Layer Two Tunneling Protocol (L2TP), no messages are exchanged for tunnel setup, maintenance, or termination. Additionally, IPv6 over IPv4 tunneling does not provide security for tunneled IPv6 packets.

### Router-to-Router

In the tunneling configuration between routers, two IPv6/IPv4 routers connect two IPv4 or IPv6 infrastructures over an IPv4 infrastructure. The tunnel spans a logical link in the path between the source and destination. The IPv6 over IPv4 tunnel between the two routers acts as a single hop. Routes within each IPv4 or IPv6 infrastructure point to the IPv6/IPv4 router on the edge. For each IPv6/IPv4 router, an interface represents the IPv6 over IPv4 tunnel and routes that use the tunnel interface.

The following figure shows tunneling between two routers.



Examples of this tunneling configuration include:

## IPv6 Technical Reference (Microsoft Corporation)

- An IPv6-only test lab that tunnels across an organization's IPv4 infrastructure to reach the IPv6 Internet.
- Two IPv6-only routing domains that tunnel across the IPv4 Internet.
- A 6to4 router that tunnels across the IPv4 Internet to reach another 6to4 router or a 6to4 relay router. For more information about "6to4", see "6to4" later in this section.

### Host-to-Router and Router-to-Host

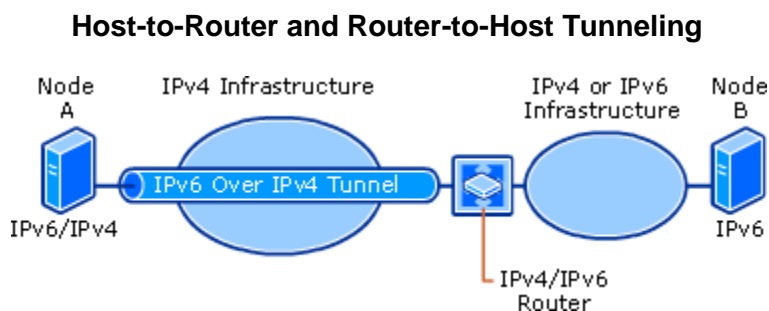
In the host-to-router tunneling configuration, an IPv6/IPv4 node that resides within an IPv4 infrastructure creates an IPv6 over IPv4 tunnel to reach an IPv6/IPv4 router. The tunnel spans the first segment of the path between the source and destination nodes. The IPv6 over IPv4 tunnel between the IPv6/IPv4 node and the IPv6/IPv4 router acts as a single hop.

On the IPv6/IPv4 node, a tunnel interface representing the IPv6 over IPv4 tunnel is created, and a route (typically a default route) is added using the tunnel interface. The IPv6/IPv4 node tunnels the IPv6 packet based on the matching route, the tunnel interface, and the next-hop address of the IPv6/IPv4 router.

In the router-to-host tunneling configuration, an IPv6/IPv4 router creates an IPv6 over IPv4 tunnel across an IPv4 infrastructure to reach an IPv6/IPv4 node. The tunnel endpoints span the last segment of the path between the source node and destination node. The IPv6 over IPv4 tunnel between the IPv6/IPv4 router and the IPv6/IPv4 node acts as a single hop.

On the IPv6/IPv4 router, a tunnel interface representing the IPv6 over IPv4 tunnel is created, and a route (typically a subnet route) is added using the tunnel interface. The IPv6/IPv4 router tunnels the IPv6 packet based on the matching subnet route, the tunnel interface, and the destination address of the IPv6/IPv4 node.

The following figure shows host-to-router (for traffic traveling from Node A to Node B) and router-to-host (for traffic traveling from Node B to Node A) tunneling.



Examples of host-to-router and router-to-host tunneling include:

- An IPv6/IPv4 host that tunnels across an organization's IPv4 infrastructure to reach the IPv6 Internet.
- An ISATAP host that tunnels across an IPv4 network to an ISATAP router to reach the IPv4 Internet, another IPv4 network, or an IPv6 network. For more information about "ISATAP", see "ISATAP" later in this section.

## IPv6 Technical Reference (Microsoft Corporation)

- An ISATAP router that tunnels across an IPv4 network to reach an ISATAP host.

# IPv6 Technical Reference

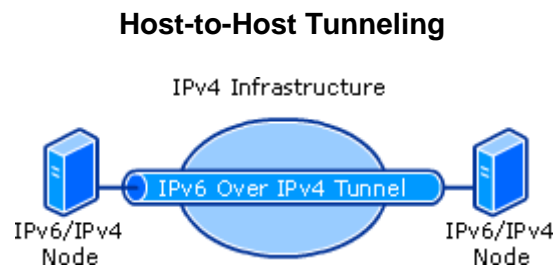
(Microsoft Corporation)

## Host-to-Host

In the tunneling configuration between hosts, an IPv6/IPv4 node that resides within an IPv4 infrastructure creates an IPv6 over IPv4 tunnel to reach another IPv6/IPv4 node that resides within the same IPv4 infrastructure. The tunnel spans the entire path between the source and destination nodes. The IPv6 over IPv4 tunnel between the IPv6/IPv4 nodes acts as a single hop.

On each IPv6/IPv4 node, an interface representing the IPv6 over IPv4 tunnel is created. Routes might indicate that the destination node is on the same logical subnet defined by the IPv4 infrastructure. Based on the sending interface, the optional route, and the destination address, the sending host tunnels the IPv6 traffic to the destination.

The following figure shows tunneling between hosts.



Examples of this tunneling configuration include:

- IPv6/IPv4 hosts that use ISATAP addresses to tunnel across an organization's IPv4 infrastructure.
- IPv6/IPv4 hosts that use IPv4-compatible addresses to tunnel across an organization's IPv4 infrastructure.

## Types of Tunnels

RFC 2893 defines the following types of tunnels:

- Configured
- Automatic

### Configured Tunnels

A configured tunnel is one whose endpoints must be configured manually. In a configured tunnel, the endpoints have IPv4 addresses that are not derived from the IPv6 source or destination addresses or the next-hop address of the matching route.

Tunnels between routers are typically configured manually. To configure the tunnel interface, the IPv4 addresses of the tunnel endpoints must be manually specified along with static routes that use the tunnel interface.

To manually configure tunnels for the IPv6 protocol for Windows Server 2003, use the `netsh interface ipv6 add v6v4tunnel` command.

### Automatic Tunnels

An automatic tunnel is one whose endpoints do not need to be configured manually. Tunnel endpoints are determined from logical tunnel interfaces, routes, and source and destination IPv6 addresses.

## IPv6 Technical Reference (Microsoft Corporation)

The IPv6 protocol for Windows Server 2003 supports the following automatic tunneling technologies:

- 6to4, which is enabled by default.
- ISATAP, which is enabled by default. For more information, see “ISATAP” later in this section.
- IPv6 Automatic Tunneling, which is disabled by default.
- 6over4, which is disabled by default.

Additionally, the IPv6 protocol for Windows XP with Service Pack 1 also supports Teredo when the Advanced Networking Pack for Windows XP is installed. Teredo is enabled by default. For more information, see “Teredo” later in this section.

### 6to4

6to4 is a technology that assigns addresses and automatically configures tunnels between routers to provide unicast IPv6 connectivity between IPv6 sites and hosts across the IPv4 Internet. 6to4 uses the global address prefix:

`2002:WWXX:YYZZ::/48`

in which WWXX:YYZZ is the colon-hexadecimal representation of a public IPv4 address (w.x.y.z) assigned to a site or host. The full 6to4 address is:

`2002:WWXX:YYZZ:[Subnet ID]:[Interface ID]`

RFC 3056 describes 6to4 in the following terms:

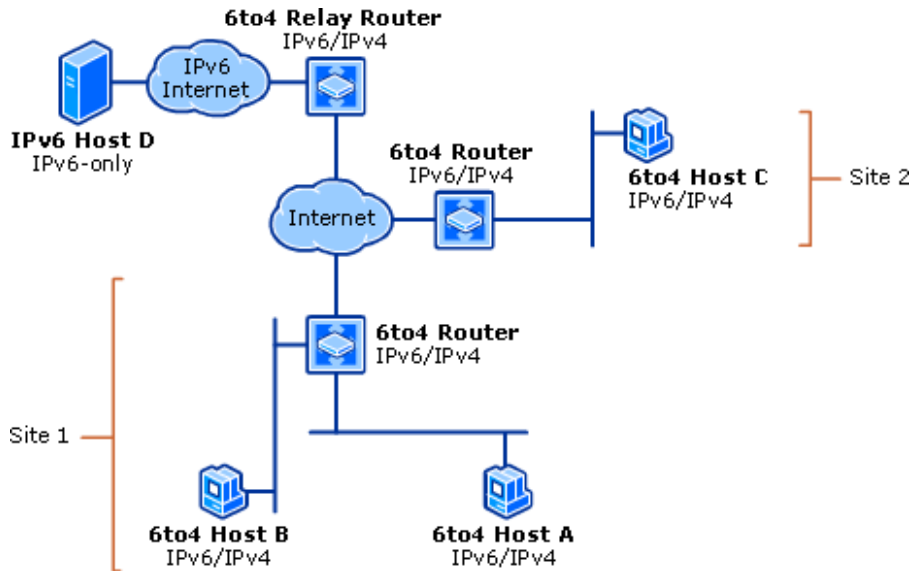
- **6to4 Host:** Any IPv6 host that is configured with at least one 6to4 address (a global address with the 2002::/16 prefix). 6to4 hosts do not require any manual configuration, and they create 6to4 addresses using standard address autoconfiguration mechanisms.
- **6to4 Router:** An IPv6/IPv4 router that supports the use of a 6to4 tunnel interface and that is typically used to forward 6to4-addressed traffic between the 6to4 hosts within a site and other 6to4 routers or 6to4 relay routers on an IPv4 internetwork, such as the Internet. 6to4 routers require additional processing logic for proper encapsulation and decapsulation, and they might require manual configuration.
- **6to4 Relay Router:** An IPv6/IPv4 router that forwards 6to4-addressed traffic between 6to4 routers on the Internet and hosts on the IPv6 Internet.

The following figure shows 6to4 components.

# IPv6 Technical Reference

## (Microsoft Corporation)

### 6to4 Components



Within a site, IPv6 routers advertise `2002:WWXX:YYZZ:[Subnet ID]::/64` prefixes so that hosts can create an autoconfigured 6to4 address and 64-bit prefix routes are used to deliver traffic between 6to4 hosts within the site. Hosts on individual subnets are automatically configured with a 64-bit subnet route for direct delivery to neighbors and a default route with the next-hop address of the advertising router. All IPv6 traffic that does not match a 64-bit prefix used by one of the subnets within the site is forwarded to a 6to4 router on the site border.

The 6to4 router on the site border has a `2002::/16` route that is used to forward traffic to other 6to4 sites and a default route (`::/0`) that is used to forward traffic to a 6to4 relay router.

In the example network shown in the previous figure, Host A and Host B can communicate with each other because of a default route using the next-hop address of the 6to4 router in Site 1. When Host A communicates with Host C in another site, Host A sends the traffic to the 6to4 router in Site 1 as IPv6 packets. The 6to4 router in Site 1, using the 6to4 tunnel interface and the `2002::/16` route in its routing table, encapsulates the packet with an IPv4 header and tunnels the packet to the 6to4 router in Site 2. When the 6to4 router in Site 2 receives the packet, the router removes the IPv4 header and, using the 64-bit prefix route in its routing table, forwards the IPv6 packet to Host C.

In this example, Host A (with the interface ID `ID_A`) resides on subnet 1 within Site 1, which uses the public IPv4 address of `157.60.91.123`. Host C (with the interface ID `ID_C`) resides on subnet 2 within Site 2, which uses the public IPv4 address of `131.107.210.49`. When the 6to4 router in Site 1 sends the IPv4-encapsulated IPv6 packet to the 6to4 router in Site 2, the addresses in the IPv4 and IPv6 headers are as listed in the following table.

### Example 6to4 Addresses

Field	Value
IPv6 Source Address	<code>2002:9D3C:5B7B:1:[ID_A]</code>
IPv6 Destination Address	<code>2002:836B:D231:2:[ID_C]</code>

## IPv6 Technical Reference (Microsoft Corporation)

IPv4 Source Address	157.60.91.123
IPv4 Destination Address	131.107.210.49

For a more detailed example of 6to4 traffic using ISATAP-derived interface identifiers, see “ISATAP” later in this section.

The following types of communication are possible when 6to4 hosts, an IPv6 routing infrastructure within a site, a 6to4 router at the site boundary, and a 6to4 relay router are used:

- A 6to4 host can communicate with another 6to4 host within the same site. This type of communication is available with the IPv6 routing infrastructure, which provides reachability to all hosts within the site. The previous figure shows this type of communication between Host A and Host B.
- A 6to4 host can communicate with 6to4 hosts in other sites across the IPv4 Internet. This type of communication occurs when a 6to4 host forwards IPv6 traffic that is destined to a 6to4 host in another site to the 6to4 router for the local site. The 6to4 router for the local site tunnels the IPv6 traffic through the IPv4 Internet to the 6to4 router at the destination site. The 6to4 router at the destination site removes the IPv4 header and forwards the IPv6 packet to the appropriate 6to4 host by using the IPv6 routing infrastructure of the destination site. The previous figure shows this type of communication between Host A and Host C.
- A 6to4 host can communicate with hosts on the IPv6 Internet. This type of communication occurs when a 6to4 host forwards IPv6 traffic that is destined for an IPv6 Internet host to the 6to4 router for the local site. The 6to4 router for the local site tunnels the IPv6 traffic to a 6to4 relay router that is connected to both the IPv4 Internet and the IPv6 Internet. The 6to4 relay router removes the IPv4 header and forwards the IPv6 packet to the appropriate IPv6 Internet host by using the routing infrastructure of the IPv6 Internet. The previous figure shows this type of communication between Host A and Host D.

All of these types of communication use IPv6 traffic without having to obtain either a direct connection to the IPv6 Internet or an IPv6 global address prefix from an ISP.

### Communicating Using a 6to4 Address

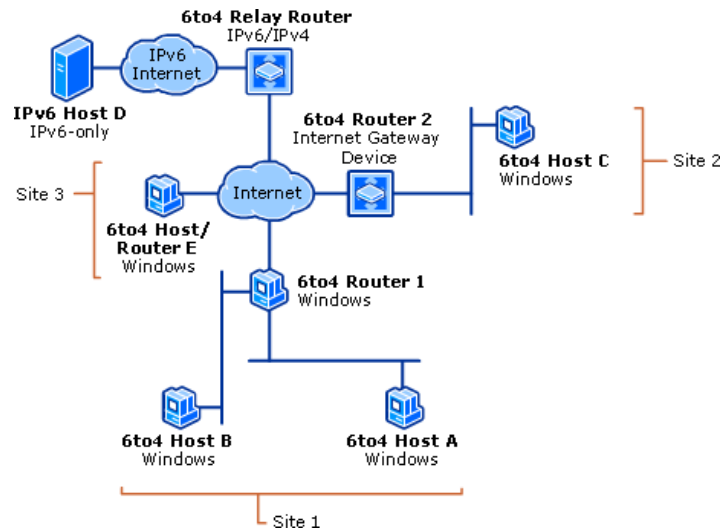
The IPv6 protocol for Windows Server 2003 contains a 6to4 component that supports 6to4 hosts and 6to4 routers. If a public IPv4 address is assigned to an interface on the host and a global prefix is not received in a router advertisement, the 6to4 component:

- Configures 6to4 addresses on the 6to4 Tunneling Pseudo-Interface for all public IPv4 addresses that are assigned to interfaces on the computer.
- Creates a 2002::/16 route that forwards all 6to4 traffic with the 6to4 Tunneling Pseudo-Interface (interface index 3). All traffic forwarded by this host to 6to4 destinations is encapsulated with an IPv4 header.
- Performs a DNS query to obtain the IPv4 address of a 6to4 relay router on the Internet. You can also use the netsh interface ipv6 6to4 set relay command to specify the DNS name to query. If the query is successful, a default route is added using the 6to4 Tunneling Pseudo-Interface, and the next-hop address is set to the 6to4 address of the 6to4 relay router.

## IPv6 Technical Reference (Microsoft Corporation)

The results of 6to4 component autoconfiguration vary depending on the configuration of the host. The following figure shows how 6to4 is configured for different types of hosts running Windows Server 2003 (except IPv6 host D).

### 6to4 for Hosts Running Windows



For a host that is assigned a private IPv4 address or receives a router advertisement for a global prefix, no 6to4 addresses are assigned to the 6to4 Tunneling Pseudo-Interface. Addresses are autoconfigured based on the global prefix, and the routing table contains a 64-bit global prefix route and default route. This configuration corresponds to Host A, Host B, and Host C in the previous figure. For a host that is assigned a public IPv4 address and does not receive a router advertisement for a global prefix, a 6to4 address of the form 2002:WWXX:YYZZ::WWXX:YYZZ is automatically assigned to the 6to4 Tunneling Pseudo-Interface. A 2002::/16 route using the 6to4 Tunneling Pseudo-Interface is added, and, if the DNS query for the 6to4 relay router is successful, a default route using the 6to4 address of the 6to4 relay router as the next hop is added. This configuration corresponds to Host E in the previous figure, a host that is directly connected to the IPv4 Internet. In this case, the host is acting as its own site and its own 6to4 router.

The 6to4 component can also enable a computer running Windows Server 2003 as a 6to4 router by utilizing the configuration of the Internet Connection Sharing (ICS) feature. This configuration corresponds to 6to4 router 1 and 6to4 router 2 in the previous figure.

If ICS is enabled on an interface that is assigned a public IPv4 address, the 6to4 component automatically:

- Enables IPv6 forwarding on both the public and private interfaces. The public interface is connected to the Internet. The private interface is connected to a single-subnet intranet and uses private IPv4 addresses from the 192.168.0.0/24 prefix.
- Sends Router Advertisement messages on the private interface. The router advertisements advertise the ICS computer as a default router and contain a global 6to4 address prefix that is based on the public IPv4 address assigned to the public interface. The Subnet ID in the 6to4 address prefix is set to the interface index of the interface on which the advertisements are sent.

## IPv6 Technical Reference (Microsoft Corporation)

For example, for an ICS computer using the public IPv4 address of 131.107.23.89 and interface 5 as the private interface, the advertised prefix would be 2002:836B:1759:5::/64. Private hosts that receive this router advertisement would create global addresses through normal address autoconfiguration. The hosts would also add a 2002:836B:1759:5::/64 route for the local subnet and a default route with a next-hop address of the link-local address of the ICS computer's private interface. Private hosts can communicate with each other on the same subnet using the 2002:836B:1759:5::/64 route. For all other 6to4 sites or the IPv6 Internet, the IPv6 packets are forwarded to the ICS computer using the default route.

For traffic to other 6to4 sites, the ICS computer uses its 2002::/16 route and encapsulates the IPv6 traffic with IPv4 headers and sends the packets across the IPv4 Internet to another 6to4 router. For all other IPv6 traffic, the ICS computer uses its default route and encapsulates the IPv6 traffic with IPv4 headers and sends the packets across the IPv4 Internet to a 6to4 relay router.

**Note:** The 6to4 component does not perform network address translation on the IPv6 packets that it forwards. ICS translates network addresses for IPv4 packets being forwarded to and from private hosts. The 6to4 component uses the ICS configuration to determine the public IPv4 address and public interface.

### ISATAP

ISATAP is a technology that assigns addresses, configures tunnels between hosts and between routers and hosts, and provides unicast IPv6 connectivity between IPv6 hosts across an IPv4 intranet. ISATAP is described in the Internet draft titled "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)." ISATAP hosts do not require any manual configuration, and they create ISATAP addresses using standard mechanisms for address autoconfiguration.

IPv6/IPv4 nodes can use ISATAP to communicate on an IPv4 network. ISATAP addresses use the locally administered interface identifier ::0:5EFE:w.x.y.z where the w.x.y.z portion is any public or private unicast IPv4 address.

The ISATAP interface identifier can be combined with any 64-bit prefix that is valid for IPv6 unicast addresses. This includes the link-local address prefix (FE80::/64) and global prefixes (including 6to4 prefixes).

Like IPv4-compatible addresses, 6over4 addresses, and 6to4 addresses, ISATAP addresses contain embedded IPv4 addresses that are used to determine either the source or destination IPv4 addresses within the IPv4 header when ISATAP-addressed IPv6 traffic is tunneled across an IPv4 network.

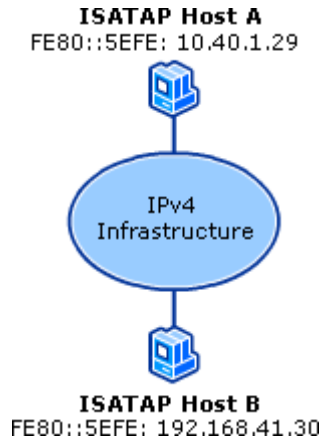
By default, the IPv6 protocol for Windows Server 2003 automatically configures the link-local ISATAP address of FE80::5EFE:w.x.y.z on the Automatic Tunneling Pseudo-Interface (interface index 2) for each IPv4 address that is assigned to the node. This link-local ISATAP address allows two hosts to communicate over an IPv4 network by using each other's link-local ISATAP addresses.

For example, Host A is configured with the IPv4 address of 10.40.1.29, and Host B is configured with the IPv4 address of 192.168.41.30. When the IPv6 protocol for Windows Server 2003 is started, Host A is automatically configured with the ISATAP address of FE80::5EFE:10.40.1.29, and Host B is automatically configured with the ISATAP address of FE80::5EFE:192.168.41.30. The following figure shows this configuration.

# IPv6 Technical Reference

(Microsoft Corporation)

## An Example ISATAP Configuration



When Host A sends IPv6 traffic to Host B by using Host B's link-local ISATAP address, the source and destination addresses for the IPv6 and IPv4 headers are as listed in the following table.

### Example Link-Local ISATAP Addresses

Field	Value
IPv6 Source Address	FE80::5EFE:10.40.1.29
IPv6 Destination Address	FE80::5EFE:192.168.41.30
IPv4 Source Address	10.40.1.29
IPv4 Destination Address	192.168.41.30

To test connectivity, use the ping command. For example, Host A would use the following command to ping Host B by using its link-local ISATAP address:

```
ping FE80::5EFE:192.168.41.30%2
```

Because the destination of the ping command is a link-local address, the %ZoneID portion of the command specifies the interface index of the interface from which traffic is sent. In this case, %2 specifies interface 2, which is the interface index assigned to the Automatic Tunneling Pseudo-Interface on Host A. The Automatic Tunneling Pseudo-Interface uses the link-local ISATAP address assigned to the interface as a source. The Automatic Tunneling Pseudo-Interface also uses the last 32 bits in the source and destination IPv6 addresses (corresponding to the embedded IPv4 addresses) as the source and destination IPv4 addresses.

### ISATAP Router

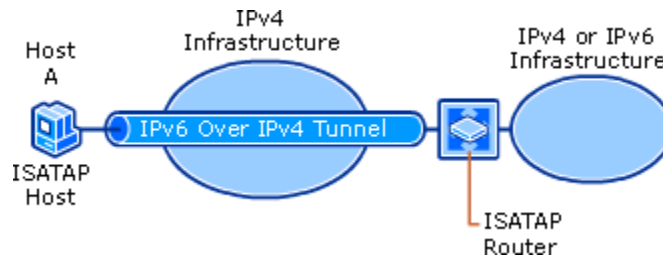
The use of link-local ISATAP addresses allows IPv6/IPv4 hosts on the same logical subnet (an IPv4 network) to communicate with each other, but not with other IPv6 addresses on other subnets. To communicate outside the logical subnet using ISATAP-derived global addresses, IPv6 hosts using ISATAP addresses must tunnel their packets to an ISATAP router. The following figure shows this configuration.

**IPv6 Technical Reference**  
(Microsoft Corporation)

# IPv6 Technical Reference

(Microsoft Corporation)

## An ISATAP Router



An ISATAP router is an IPv6 router that:

- Forwards packets between ISATAP hosts on a logical subnet (an IPv4 network) and hosts on other subnets. The other subnets can be other IPv4 networks (such as a portion of an organization network or the IPv4 Internet) or subnets in a native IPv6 routing domain (such as an organization's IPv6 network or the IPv6 Internet).
- Acts as a default router for ISATAP hosts.
- Advertises address prefixes to identify the logical subnet on which ISATAP hosts are located. ISATAP hosts use the advertised address prefixes to configure global ISATAP addresses.

When an ISATAP host receives a Router Advertisement message from an ISATAP router, a default route (::/0) is added using the Automatic Tunneling Pseudo-Interface with next-hop address set to the link-local ISATAP address that corresponds to the logical subnet interface of the ISATAP router. When packets destined to locations outside the logical subnet are sent, they are tunneled to the IPv4 address that corresponds to the ISATAP router's interface on the logical subnet defined by the IPv4 network containing the ISATAP router and ISATAP host. The ISATAP router then forwards the IPv6 packet.

For the IPv6 protocol for Windows Server 2003, the configuration of the intranet IPv4 address of the ISATAP router is obtained through one of the following:

- The resolution of the name ISATAP to an IPv4 address.
- The netsh interface ipv6 isatap set router command.

### Resolving the ISATAP Name

When the IPv6 protocol for Windows Server 2003 starts, it attempts to resolve the name "ISATAP" to an IPv4 address using normal TCP/IP name resolution techniques. These techniques include the following:

1. Checking the local host name.
2. Checking the DNS client resolver cache, which includes the entries in the Hosts file. This file is located in the systemroot\system32\drivers\etc folder.
3. Forming a fully qualified domain name and sending a DNS name query. For example, if the computer running Windows Server 2003 is a member of the example.microsoft.com domain (and example.microsoft.com is the only domain name in the search list), the computer sends a DNS query to resolve the name isatap.example.microsoft.com.

## IPv6 Technical Reference (Microsoft Corporation)

4. Converting the ISATAP name into the NetBIOS name "ISATAP <00>" and checking the NetBIOS name cache.
5. Sending a NetBIOS name query to the configured Windows Internet Name Service (WINS) servers.
6. Sending NetBIOS broadcasts.
7. Checking the Lmhosts file. This file is located in the systemroot\system32\drivers\etc folder.

If the name is resolved, the host sends an IPv4-encapsulated Router Solicitation message to the ISATAP router. The ISATAP router responds with an IPv4-encapsulated unicast Router Advertisement message that contains prefixes to use for autoconfiguring ISATAP-based addresses and, optionally, advertising itself as a default router.

To ensure that at least one of these attempts is successful, you can do one of the following:

- If the ISATAP router is a computer running Windows Server 2003, name the computer ISATAP, and it will automatically register the appropriate records in DNS and WINS.
- Manually create an ISATAP address (A) record in the appropriate domain in DNS. For example, create an A record for isatap.example.microsoft.com.
- Manually create a static WINS record in WINS for the NetBIOS name "ISATAP <00>."
- Add the following entry to the Hosts file of the computers that need to resolve the name ISATAP:  
IPv4Address ISATAP
- Add the following entry to the Lmhosts file of the computers that need to resolve the name ISATAP:  
ISATAP: IPv4Address ISATAP

### **Resolving the \_ISATAP Name for Windows XP (without SP1)**

When the IPv6 protocol for Windows XP (without SP1) starts, it attempts to resolve the name "\_ISATAP," rather than "ISATAP." To ensure that a computer running Windows XP (without SP1) can resolve the name \_ISATAP, you can do one of the following:

- Manually create \_ISATAP canonical name (CNAME) records in the appropriate domains in DNS. A CNAME record maps a name that is an alias to another name. For example, assuming that an A record already exists for the name isatap.example.microsoft.com, create a CNAME record that maps \_isatap.example.microsoft.com to isatap.example.microsoft.com.
- Manually create a static WINS record for the NetBIOS name "\_ISATAP <00>."
- Add the following entry to the Hosts file of the computers running Windows XP: IPv4Address \_ISATAP
- Add the following entry to the Lmhosts file of the computers running Windows XP: IPv4Address \_ISATAP

## IPv6 Technical Reference (Microsoft Corporation)

Note: Windows XP with SP1 attempts to resolve the name "ISATAP" to determine the IPv4 address of the ISATAP router. The methods described here are not needed if all your computers are running either Windows Server 2003 or Windows XP with SP1.

### Using the Netsh Interface Ipv6 Isatap Set Router Command

Although the automatic resolution of the ISATAP name is the recommended method for configuring the IPv4 address of the ISATAP router, you can configure this address manually by using the netsh interface ipv6 isatap set router command. The syntax of this command is:

```
netsh interface ipv6 isatap set router AddressOrName
```

where AddressOrName is the name or IPv4 address of the ISATAP router's intranet interface. For example, if the ISATAP router's IPv4 address is 192.168.39.1, the command is:

```
netsh interface ipv6 isatap set router 192.168.39.1
```

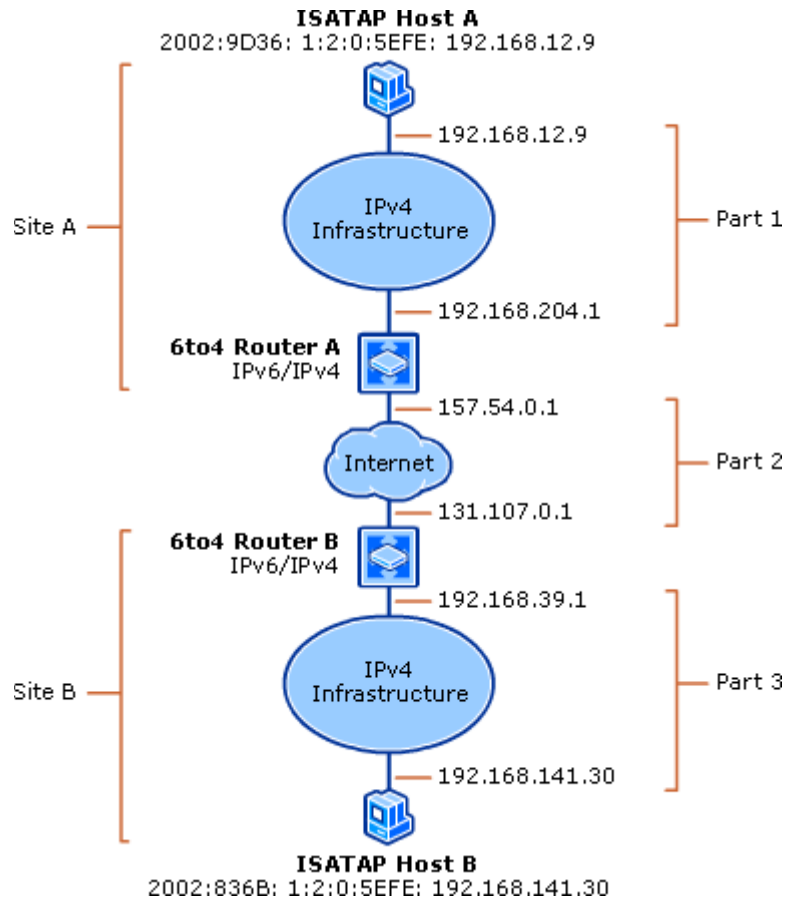
After a host has been configured, it sends an IPv4-encapsulated Router Solicitation message to the ISATAP router. The ISATAP router responds with an IPv4-encapsulated unicast Router Advertisement message that contains prefixes to use for autoconfiguring ISATAP-based addresses. This additional configuration is needed only when no IPv6 router is on the host's subnet.

### ISATAP and 6to4 Example

The following figure shows two ISATAP hosts that are using 6to4 prefixes to communicate across the Internet even though each site is using the 192.168.0.0/16 private address space internally.

#### Communication Between ISATAP Hosts in Different 6to4 Sites

## IPv6 Technical Reference (Microsoft Corporation)



In this configuration:

- ISATAP Host A automatically configures a link-local ISATAP address of FE80::5EFE:192.168.12.9 on its Automatic Tunneling Pseudo-Interface.
- 6to4 Router A automatically configures a link-local ISATAP address of FE80::5EFE:192.168.204.1 on its Automatic Tunneling Pseudo-Interface.
- 6to4 Router B automatically configures a link-local ISATAP address of FE80::5EFE:192.168.39.1 on its Automatic Tunneling Pseudo-Interface.
- ISATAP Host B automatically configures a link-local ISATAP address of FE80::5EFE:192.168.141.30 on its Automatic Tunneling Pseudo-Interface.

ISATAP Host A can reach 6to4 Router A and all other hosts within Site A using link-local ISATAP addresses. However, ISATAP Host A cannot reach any addresses outside Site A. 6to4 Router A constructs the global prefix 2002:9D36:1:5::/64. (9D36:1 is the colon-hexadecimal notation for 157.54.0.1 and 5 is the interface index of 6to4 Router A's intranet interface.) 6to4 Router A also advertises the prefix using a Router Advertisement message on its intranet interface. However, ISATAP Host A is not on 6to4 Router A's subnet and will never create a global address based on this 6to4 prefix.

To configure ISATAP Host A to receive the router advertisement from 6to4 Router A, the network administrator for Site A has configured 6to4 Router A as an ISATAP router. The administrator has

## IPv6 Technical Reference (Microsoft Corporation)

also added an A record to Site A's DNS infrastructure so that the name ISATAP is resolved to the IPv4 address of 192.168.204.1. Upon startup, the IPv6 protocol on Host A resolves the ISATAP name and sends a Router Solicitation message to the addresses listed in the following table.

### Addresses in the Router Solicitation Message

Field	Value
IPv6 Source Address	FE80::5EFE:192.168.12.9
IPv6 Destination Address	FF02::2
IPv4 Source Address	192.168.12.9
IPv4 Destination Address	192.168.204.1

6to4 Router A receives the Router Solicitation message from ISATAP Host A and sends back a unicast Router Advertisement message advertising itself as a default router with a Prefix Information option to automatically configure IPv6 addresses using the prefix 2002:9D36:1:2::/64. (9D36:1 is the colon-hexadecimal notation for 157.54.0.1, and 2 is the interface index of 6to4 Router A's Automatic Tunneling Pseudo-Interface.)

The Router Advertisement message is sent to the addresses listed in the following table.

### Addresses in the Router Advertisement Message

Field	Value
IPv6 Source Address	FE80::5EFE:192.168.204.1
IPv6 Destination Address	FE80::5EFE:192.168.12.9
IPv4 Source Address	192.168.204.1
IPv4 Destination Address	192.168.12.9

ISATAP Host A receives the Router Advertisement message and autoconfigures the address 2002:9D36:1:2:0:5EFE:192.168.12.9. The host also configures a default route (::/0) using the Automatic Tunneling Pseudo-Interface (interface index 2) with the next-hop address of FE80::5EFE:192.168.204.1 and a 2002:9D36:1:2::/64 route using the Automatic Tunneling Pseudo-Interface.

Similarly, 6to4 Router B is configured as an ISATAP router, and Site B has an appropriate A record in its DNS infrastructure so that ISATAP Host B autoconfigures the address 2002:836B:1:2:0:5EFE:192.168.141.30. (836B:1 is the colon-hexadecimal notation for 131.107.0.1.) ISATAP Host B also configures a default route (::/0) using the Automatic Tunneling Pseudo-Interface (interface index 2) with the next-hop address of FE80::5EFE:192.168.39.1 and a 2002:836B:1:2::/64 route using the Automatic Tunneling Pseudo-Interface.

ISATAP Host A can now send a packet to ISATAP B. Packet addressing comprises three parts (as shown in the previous figure during the packet's trip from ISATAP Host A to ISATAP Host B).

### Part 1: from ISATAP Host A to 6to4 Router A

## IPv6 Technical Reference (Microsoft Corporation)

ISATAP Host A sends the IPv6 packet with the ::/0 route that uses the Automatic Tunneling Pseudo-Interface to the next-hop address of FE80::5EFE:192.168.204.1. By using this route, the next-hop address for this packet is set to the link-local ISATAP address of 6to4 Router A (FE80::5EFE:192.168.204.1).

Using the Automatic Tunneling Pseudo-Interface, the packet is tunneled using IPv4 from the address assigned to its intranet interface (192.168.12.9) to the embedded address in the ISATAP interface ID of the next-hop address (192.168.204.1). The resulting addresses are listed in the following table.

**Addresses in Part 1**

Field	Value
IPv6 Source Address	2002:9D36:1:2:0:5EFE:192.168.12.9
IPv6 Destination Address	2002:836B:1:2:0:5EFE:192.168.141.30
IPv4 Source Address	192.168.12.9
IPv4 Destination Address	192.168.204.1

### **Part 2: from 6to4 Router A to 6to4 Router B**

6to4 Router A receives the IPv4 packet and removes the IPv4 header. 6to4 Router A then forwards the IPv6 packet with the 2002::/16 route that uses the 6to4 Tunneling Pseudo-Interface. The router, by using this route, ensures that the next-hop address for this packet is set to the destination address (2002:836B:1:2:0:5EFE:192.168.141.30).

The packet is tunneled using IPv4 and the 6to4 Tunneling Pseudo-Interface from the address assigned to its Internet interface (157.54.0.1) to the embedded address in the 6to4 prefix (836B:1) of the next-hop address (131.107.0.1). The resulting addresses are listed in the following table.

**Addresses in Part 2**

Field	Value
IPv6 Source Address	2002:9D36:1:2:0:5EFE:192.168.12.9
IPv6 Destination Address	2002:836B:1:2:0:5EFE:192.168.141.30
IPv4 Source Address	157.54.0.1
IPv4 Destination Address	131.107.0.1

### **Part 3: from 6to4 Router B to ISATAP Host B**

6to4 Router B receives the IPv4 packet and removes the IPv4 header. 6to4 Router B then forwards the IPv6 packet with the 2002:836B:1:2::/64 route that uses the ISATAP Automatic Tunneling Pseudo-Interface. The router, by using this route, ensures that the next-hop address for this packet is set to the destination address (2002:836B:1:2:0:5EFE:192.168.141.30).

Because the packet is forwarded using Automatic Tunneling Pseudo-Interface, the packet is tunneled using IPv4 from the address assigned to its intranet interface (192.168.39.1) to the embedded address in the ISATAP interface ID of the next-hop IPv6 address (192.168.141.30). 6to4 Router B sets the addresses in the forwarded packet as listed in the following table.

# IPv6 Technical Reference

(Microsoft Corporation)

## Addresses in Part 3

Field	Value
IPv6 Source Address	2002:9D36:1:2:0:5EFE:192.168.12.9
IPv6 Destination Address	2002:836B:1:2:0:5EFE:192.168.141.30
IPv4 Source Address	192.168.39.1
IPv4 Destination Address	192.168.141.30

### PortProxy

To facilitate communication between nodes or applications that cannot connect using a common Internet layer protocol (IPv4 or IPv6), the IPv6 protocol for Windows Server 2003 provides PortProxy, a component that allows the following traffic to be proxied:

- **IPv4 to IPv4:** TCP traffic to an IPv4 address is proxied to TCP traffic to another IPv4 address.
- **IPv4 to IPv6:** TCP traffic to an IPv4 address is proxied to TCP traffic to an IPv6 address.
- **IPv6 to IPv6:** TCP traffic to an IPv6 address is proxied to TCP traffic to another IPv6 address.
- **IPv6 to IPv4:** TCP traffic to an IPv6 address is proxied to TCP traffic to an IPv4 address. The most interesting and useful proxying for IPv6/IPv4 coexistence and migration is from IPv4 to IPv6 and from IPv6 to IPv4. For coexistence and migration, PortProxy enables the following scenarios:
  - An IPv4-only node can access an IPv6-only node. In the IPv4 DNS infrastructure of the IPv4-only node, the name of the IPv6-only node resolves to an IPv4 address assigned to an interface of the PortProxy computer. (Resolving this name might require an A record to be configured manually in DNS.) The PortProxy computer is configured to proxy IPv4 to IPv6. All TCP traffic sent by the IPv4-only node is proxied in a manner similar to Internet proxy servers: the IPv4-only node establishes a TCP connection with the PortProxy computer, and the PortProxy computer establishes a separate TCP connection with the IPv6-only node. The TCP connection data is transferred between the IPv4-only node and the IPv6-only node by the PortProxy component.
  - An IPv6-only node can access an IPv4-only node. In the IPv6 DNS infrastructure of the IPv6-only node, the name of the IPv4-only node resolves to an IPv6 address assigned to an interface of the PortProxy computer. (Resolving this name might require AAAA records to be manually configured in DNS.) The PortProxy computer is configured to proxy IPv6 to IPv4. TCP traffic sent by the IPv6-only node to the PortProxy computer is proxied to the IPv4-only node.
  - An IPv6 node can access an IPv4-only service running on a PortProxy computer. In the IPv6 DNS infrastructure of the IPv6-only node, the name of the IPv6/IPv4 node resolves to an IPv6 address assigned to an interface of the PortProxy computer. The PortProxy computer is configured to proxy from IPv6 to IPv4 on itself. TCP traffic sent by the IPv6 node to the PortProxy computer is proxied to an IPv4-only service or application running on the PortProxy computer.

The last scenario allows IPv6 nodes to access services that are running on a server that does not support IPv6. For example, Windows Server 2003 includes Telnet Client, which supports IPv6, and

## IPv6 Technical Reference (Microsoft Corporation)

Telnet Server, which does not. However, you can allow IPv6 nodes to access the Telnet service by running PortProxy on the computer that is running Telnet Server.

To configure the PortProxy component, use the netsh interface portproxy add|set|delete v4tov4|v4tov6|v6tov4|v6tov6 commands. For example, use the netsh interface portproxy add v6tov4 23 command to enable IPv6 on the Telnet service (using TCP port 23) on a computer that is running Windows Server 2003.

By default, the IPv6/IPv4 node that is running Telnet Server dynamically registers both its IPv6 and IPv4 addresses in DNS. By default, a computer that is running Windows Server 2003 queries DNS for all record types, preferring IPv6 addresses to IPv4 addresses. When the Telnet client is a computer that is running Windows Server 2003, it attempts to connect using IPv6 first. With PortProxy properly configured, the first attempt to connect using an IPv6 address of the computer that is running Telnet Server should succeed without manual configuration of DNS records.

**Note:** The PortProxy component is provided only with the IPv6 protocol for Windows Server 2003. The PortProxy component works only for TCP traffic and for application-layer protocols that do not embed address or port information inside the upper-layer PDU. PortProxy has no facilities to check for and change embedded address or port information in upper layer PDUs that are being proxied. For example, PortProxy cannot enable the FTP server service for IPv6 because the FTP PORT command embeds IPv4 address information inside the FTP PDU.

### IPv6 Automatic Tunneling

RFC 2893 defines IPv6 Automatic Tunneling as the tunneling that occurs when IPv4-compatible addresses (::w.x.y.z where w.x.y.z is a public IPv4 address) are used. IPv6 Automatic Tunneling is a host-to-host tunnel between two IPv6/IPv4 hosts using IPv4-compatible addresses.

For example, when Host1 (with the public IPv4 address of 157.60.91.123 and corresponding IPv4-compatible address of ::157.60.91.123) sends traffic to Host2 (with the IPv4 address of 131.107.210.49 and corresponding IPv4-compatible address of ::131.107.210.49), the addresses in the IPv4 and IPv6 headers are as listed in the following table.

**Example IPv6 Automatic Tunneling Addresses**

Field	Value
IPv6 Source Address	::157.60.91.123
IPv6 Destination Address	::131.107.210.49
IPv4 Source Address	157.60.91.123
IPv4 Destination Address	131.107.210.49

To test connectivity, use the ping command. For example, Host A would use the following command to ping Host B by using its IPv4-compatible address:

```
ping ::131.107.210.49
```

The IPv6 protocol for Windows Server 2003 does not use IPv4-compatible addresses by default. To enable IPv4-compatible addresses, use the netsh interface ipv6 set state v4compat=enabled command. When the IPv6 protocol for Windows Server 2003 is enabled, communication to IPv4-

## IPv6 Technical Reference

(Microsoft Corporation)

compatible addresses is facilitated by a `::/96` route in the IPv6 routing table that uses the Automatic Tunneling Pseudo-Interface (interface index 2). This route indicates that all packets with destination addresses in which the first 96 bits are set to 0 are forwarded to their destination addresses with the Automatic Tunneling Pseudo-Interface. The Automatic Tunneling Pseudo-Interface uses the last 32 bits in the source and destination IPv6 addresses (corresponding to the embedded IPv4 addresses) as the source and destination IPv4 addresses for the outgoing IPv4 packet.

**Note:** In this section, the term “IPv6 Automatic Tunneling” refers to the use of IPv4-compatible addresses. The term “automatic tunneling” is tunneling that occurs without manual configuration, independent of the type of addressing being used. IPv4-compatible addresses are not widely used because they are defined only for public IPv4 addresses, and their functionality has been replaced with ISATAP for IPv4 intranets and 6to4 for the IPv4 Internet. For more information, see “ISATAP” and “6to4” earlier in this section.

### 6over4

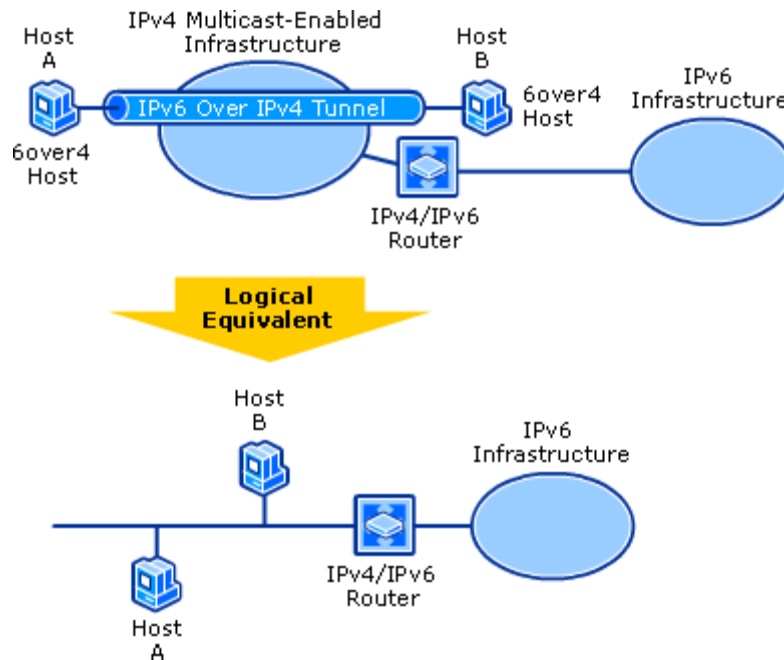
6over4, also known as IPv4 multicast tunneling, is a technology that automatically configures tunnels between hosts, between routers, and between hosts and routers. 6over4 provides unicast and multicast connectivity between IPv6 nodes across an IPv4 intranet. RFC 2529 describes 6over4. 6over4 hosts use a valid 64-bit prefix for unicast addresses and the interface identifier `::WWXX:YYZZ`, where `WWXX:YYZZ` is the colon-hexadecimal representation of the IPv4 address (`w.x.y.z`) that is assigned to the host. By default, 6over4 hosts automatically configure the link-local address `FE80::WWXX:YYZZ` on each 6over4 interface.

6over4 treats an IPv4 infrastructure as a single link with multicast capabilities. Neighbor Discovery processes (such as address resolution and router discovery) work over a physical link with multicast capabilities. To emulate a multicast-capable link, the IPv4 infrastructure must be enabled for IPv4 multicast traffic. The following figure shows a 6over4 configuration.

# IPv6 Technical Reference

(Microsoft Corporation)

## A 6over4 Configuration



To facilitate IPv6 multicast communication over an infrastructure that is enabled for IPv4 multicast traffic, RFC 2529 defines the following mapping to translate an IPv6 multicast address to an IPv4 multicast address:

`239.192.[second to last byte of IPv6 address].[last byte of IPv6 address]`

The following are example mappings for IPv6 multicast addresses:

- FF02::1 (link-local scope all-hosts multicast address) is mapped to 239.192.0.1.
- FF02::2 (link-local scope all-routers multicast address) is mapped to 239.192.0.2.
- FF02::1:FF28:9C5A (example solicited-node multicast address) is mapped to 239.192.156.90.

When 6over4 is enabled, the IPv4 layer uses Internet Group Membership Protocol (IGMP) messages to inform local IPv4 routers that it will receive IPv4 multicast traffic that is sent to the mapped IPv4 multicast addresses. Hosts that are enabled for 6over4 also register additional multicast MAC addresses with their network adapters that correspond to the mapped IPv4 multicast addresses. For example, for an Ethernet adapter:

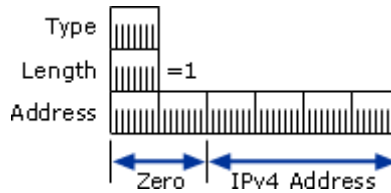
- The corresponding multicast MAC address for 239.192.0.1 is 01-00-5E-40-00-01.
- The corresponding multicast MAC address for 239.192.0.2 is 01-00-5E-40-00-02.
- The corresponding multicast MAC address for 239.192.156.90 is 01-00-5E-40-9C-5A.

Because the IPv4 infrastructure acts as a multicast-capable link, hosts can use Neighbor Solicitation and Neighbor Advertisement messages to resolve each other's link-layer addresses. The 6over4 link-layer addresses are the tunnel endpoints. Hosts and routers can use Router Solicitation and Router Advertisement messages for router, prefix, and parameter discovery. The following figure shows the format for the Source and Target Link-Layer Address options that, as defined in RFC 2529, facilitate Neighbor Discovery.

# IPv6 Technical Reference

(Microsoft Corporation)

## Source and Target Link-Layer Address Options for 6over4



For example, Host1 has the public IPv4 address of 157.60.91.123 and corresponding link-local 6over4 address of FE80::9D3C:5B7B. Host2 has the public IPv4 address of 131.107.210.49 and corresponding link-local 6over4 address of FE80::836B:D231. When Host 1 sends traffic to Host 2, the addresses in the IPv4 and IPv6 headers are as listed in the following table.

### Example 6over4 Addresses

Field	Value
IPv6 Source Address	FE80::9D3C:5B7B
IPv6 Destination Address	FE80::836B:D231
IPv4 Source Address	157.60.91.123
IPv4 Destination Address	131.107.210.49

The use of 6over4 for the IPv6 protocol for Windows Server 2003 is disabled by default. To enable the use of 6over4, use the **netsh interface ipv6 set state 6over4=enabled** command. This command creates a 6over4 tunneling interface for each IPv4 address assigned to the computer. If the computer receives Router Advertisement messages over any of these interfaces (through the multicast mapping mechanism described earlier), appropriate addresses for this interface and routes that use this interface are automatically configured.

Routes and the 6over4 tunneling interface facilitate communication to 6over4 addresses. For example, the interface index for the 6over4 tunneling interface of a host is 5. (The actual interface index for the 6over4 tunneling interface varies depending on the configuration of the computer.) A Router Advertisement message from a router with the link-local 6over4 address of FE80::C0A8:1501 is received. The router is advertised as a default router, and the message contains the auto-configuration prefix FEC0:0:0:21A8::/64. The host configures a default route with the next-hop address of FE80::C0A8:1501 and a subnet route for prefix FEC0:0:0:21A8::/64 that uses interface index 5.

When packets are sent using the default or FEC0:0:0:21A8::/64 routes, the sending host uses the appropriate locally assigned 6over4 address as a source. The node also uses the last 32 bits in the source and destination IPv6 addresses (corresponding to the embedded IPv4 addresses) as the source and destination IPv4 addresses for the outgoing IPv4 packet.

A packet to a destination that matches the prefix FEC0:0:0:21A8::/64 is sent to the next-hop address of the destination using the 6over4 tunneling interface. The 6over4 tunneling interface uses address resolution for the destination address to determine the source and destination link-layer addresses (and corresponding IPv4 addresses) to use when sending the IPv4-encapsulated IPv6 packet.

## IPv6 Technical Reference (Microsoft Corporation)

A packet to a destination that matches the default route is sent to the next-hop address of FE80::C0A8:1501 using the 6over4 tunneling interface. The 6over4 tunneling interface uses address resolution for the next-hop address to determine the source and destination link-layer addresses (and corresponding IPv4 addresses) to use when sending the IPv4-encapsulated IPv6 packet.

**Note:** 6over4 is not widely used because it requires an IPv4 multicast infrastructure. The difference between using ISATAP versus 6over4 on an IPv4 intranet is that 6over4 supports IPv6 multicast, and ISATAP currently does not. When you enable 6over4 with the `netsh interface ipv6 set state 6over4=enabled` command, it creates a 6over4 tunneling interface with a name that is based on a globally unique identifier (GUID). To create a 6over4 tunneling interface with a friendlier name, use the `netsh interface ipv6 add 6over4tunnel` command instead.

### Teredo

Teredo, also known as IPv4 network address translation (NAT) traversal for IPv6, is an IPv6/IPv4 transition technology. Teredo provides address assignment and host-to-host automatic tunneling for unicast IPv6 connectivity when IPv6/IPv4 hosts are located behind one or multiple IPv4 NATs. To traverse IPv4 NATs, IPv6 packets are sent as IPv4-based User Datagram Protocol (UDP) messages. This section provides an overview of Teredo — including Teredo addresses and packet structures — and detailed explanations of how communication is initiated between Teredo clients, Teredo host-specific relays, and IPv6-only hosts that use the IPv4 Internet, the IPv6 Internet, Teredo servers, and Teredo relays.

Teredo is an address assignment and automatic tunneling technology that provides unicast IPv6 connectivity across the IPv4 Internet. 6to4 is a well-defined automatic tunneling technology that provides unicast IPv6 connectivity across the IPv4 Internet. However 6to4 works best when a 6to4 router exists at the edge of the site. The 6to4 router uses a public IPv4 address to construct the 6to4 prefix and acts as an IPv6 advertising and forwarding router. The 6to4 router encapsulates and decapsulates IPv6 traffic sent to and from site nodes.

6to4 relies on the configuration of a public IPv4 address and the implementation of 6to4 routing functionality in the edge device. Many small office/home office (SOHO) configurations include an IPv4 network address translator (NAT). For more information about how network address translation works, see “Overview of Network Address Translators (NATs)” later in this section. In most NAT configurations, the device providing NAT functionality is not capable of being a 6to4 router. Even if all NAT devices could support 6to4, some configurations contain multiple levels of NATs. In these configurations, a 6to4-capable NAT device will not work because it does not have a public IPv4 address.

Teredo fills the need for 6to4 functionality in modern-day NATs and multi-layered NAT configurations by tunneling IPv6 packets between the hosts within the sites. In contrast, 6to4 uses tunneling from the edge device. Tunneling between hosts presents another issue for NATs: IPv4-encapsulated IPv6 packets are sent with the Protocol field in the IPv4 header set to 41. Most NATs translate only TCP or UDP traffic, and they must either be manually configured to translate other protocols or have installed NAT editors that handle the translation. Because Protocol 41 translation is not a common feature of NATs, IPv4-encapsulated IPv6 traffic will not traverse typical NATs. Therefore, to allow IPv6 traffic to traverse one or multiple NATs, the IPv6 packet is encapsulated as an IPv4 UDP message, containing both an IPv4 and a UDP header. UDP messages can be translated universally by NATs and can traverse multiple layers of NATs.

To summarize, Teredo is an IPv6/IPv4 transition technology that allows automatic IPv6 tunneling between hosts that are located across one or more IPv4 NATs. IPv6 traffic from Teredo hosts can

## IPv6 Technical Reference

(Microsoft Corporation)

traverse NATs because it is sent as IPv4 UDP messages. If a NAT supports UDP port translation, then the NAT supports Teredo. The exception is a symmetric NAT. For more information, see “Types of NATs” later in this section.

Teredo is designed as a last-resort transition technology for IPv6 connectivity. If native IPv6, 6to4, or ISATAP connectivity is present, the host does not act as a Teredo client. As more IPv4 NATs are upgraded to support 6to4 and IPv6 connectivity becomes ubiquitous, Teredo will be used less and less and eventually eliminated.

### Network Address Translators

As defined in RFC 1631, a network address translator (NAT) is an IPv4 router that can translate the IP addresses and TCP/UDP port numbers of packets as it forwards them. For example, consider a small business network with multiple computers that connect to the Internet. Without a NAT, this business would need to obtain a public IP address for each computer on the network. With a NAT, however, the small business can use private addressing (as described in RFC 1918) and have the NAT map its private addresses to a single or to multiple public IP addresses.

NAT is a common solution for the following combination of requirements:

- The administrator wants to leverage a single connection over multiple computers, rather than connecting each one to the Internet.
- The administrator wants to use private addressing.
- The administrator wants to allow access to Internet resources without having to deploy a proxy server.

### How Network Address Translation Works

When a private user on the small business intranet connects to an Internet resource, the user's TCP/IP protocol creates an IP packet with the following values set in the IP and TCP or UDP headers (bold text indicates the fields that are affected by the NAT):

- Destination IP Address: IP address of the Internet resource
- Source IP Address: Private IP address
- Destination Port: TCP or UDP port of the Internet resource
- Source Port: Source application TCP or UDP port

The source host or another router forwards this IP packet to the NAT, which translates the addresses of the outgoing packet as follows:

- Destination IP Address: IP address of the Internet resource
- Source IP Address: ISP-allocated public address
- Destination Port: TCP or UDP port of the Internet resource
- Source Port: Remapped source application TCP or UDP port

The NAT sends the remapped IP packet over the Internet. The responding computer sends back a response to the NAT. The response contains the following addressing information:

- Destination IP Address: ISP-allocated public address
- Source IP Address: IP address of the Internet resource
- Destination Port: Remapped source application TCP or UDP port

## IPv6 Technical Reference (Microsoft Corporation)

- Source Port: TCP or UDP port of the Internet resource

When the NAT maps and translates the addresses and forwards the packet to the intranet client, the packet contains the following addressing information:

- Destination IP Address: Private IP address
- Source IP Address: IP address of the Internet resource
- Destination Port: Source application TCP or UDP port
- Source Port: TCP or UDP port of the Internet resource

For outgoing packets, the source IP address and TCP/UDP port numbers are mapped to a public source IP address and a possibly changed TCP/UDP port number. For incoming packets, the destination IP address and TCP/UDP port numbers are mapped to the private IP address and original TCP/UDP port numbers.

For example, a small business is using the 192.168.0.0/24 private network ID for its intranet and has been allocated a single public IP address of 131.107.0.1 by its ISP. When a user with the private address 192.168.0.99 on the small business intranet connects to a Web server at the IP address 157.60.0.1, the user's TCP/IP protocol creates an IP packet with the following values set in the IP and TCP or UDP headers:

- Destination IP Address: 157.60.0.1
- Source IP Address: 192.168.0.99
- Destination Port: 80
- Source Port: 1025

The source host forwards this packet to the NAT device, which translates the addresses of the outgoing packet as follows:

- Destination IP Address: 157.60.0.1
- Source IP Address: 131.107.0.1
- Destination Port: 80
- Source Port: 5000

The NAT sends the remapped packet over the Internet. The Web server sends back a response to the NAT. The response contains the following addressing information:

- Destination IP Address: 131.107.0.1
- Source IP Address: 157.60.0.1
- Destination Port: 5000
- Source Port: 80

When the NAT maps and translates the addresses and forwards the packet to the intranet client, the packet contains the following addressing information:

- Destination IP Address: 192.168.0.99
- Source IP Address: 157.60.0.1
- Destination Port: 1025
- Source Port: 80

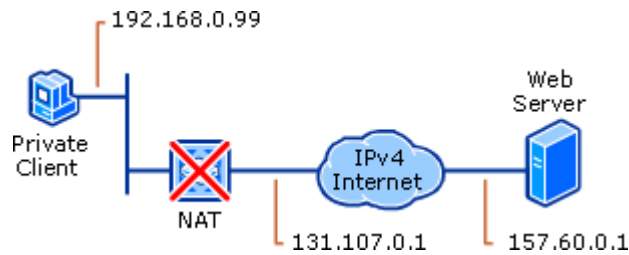
The following figure shows the configuration for this example.

**IPv6 Technical Reference**  
(Microsoft Corporation)

# IPv6 Technical Reference

(Microsoft Corporation)

## NAT Example



The mappings for private to public traffic are stored in a NAT translation table, which can contain two types of entries:

- **Dynamic Mappings:** Created when private network clients initiate communication. Dynamic mappings are removed from the NAT translation table after a specified amount of time, unless traffic that corresponds to the mapping refreshes it.
- **Static Mappings:** Configured manually so that communication that Internet clients initiate can be mapped to specific private network addresses and ports. Static mappings are needed when you want to make servers (for example, Web servers) or applications (for example, games) on the private network available to computers that are connected to the Internet. Static mappings are not automatically removed from the NAT translation table.

The NAT forwards traffic from the Internet to the private network only if the NAT translation table contains an appropriate mapping. In this way, the NAT provides a level of protection for computers that are connected to private network segments. However, a NAT should not be used in place of a fully featured firewall when Internet security is a concern.

## Types of NATs

NATs fall into the following types:

- **Cone NATs:** A NAT in which the NAT translation table entry stores a mapping between an internal address and port number and an external address and port number. When the NAT translation table entry is in place, inbound traffic to the external address and port number from any source address and port number is translated.
- **Restricted NATs:** A NAT in which the NAT translation table entry stores a mapping between an internal address and port number and an external address and port number, for either specific source addresses or specific source address and port numbers. An inbound packet from an unknown external address or port number is silently discarded.
- **Symmetric NATs:** A NAT that maps the same internal address and port number to different external addresses and ports, depending on the external destination address (for outbound traffic).

Teredo can work over only cone and restricted NATs. Teredo cannot work over symmetric NATs.

## Teredo Components

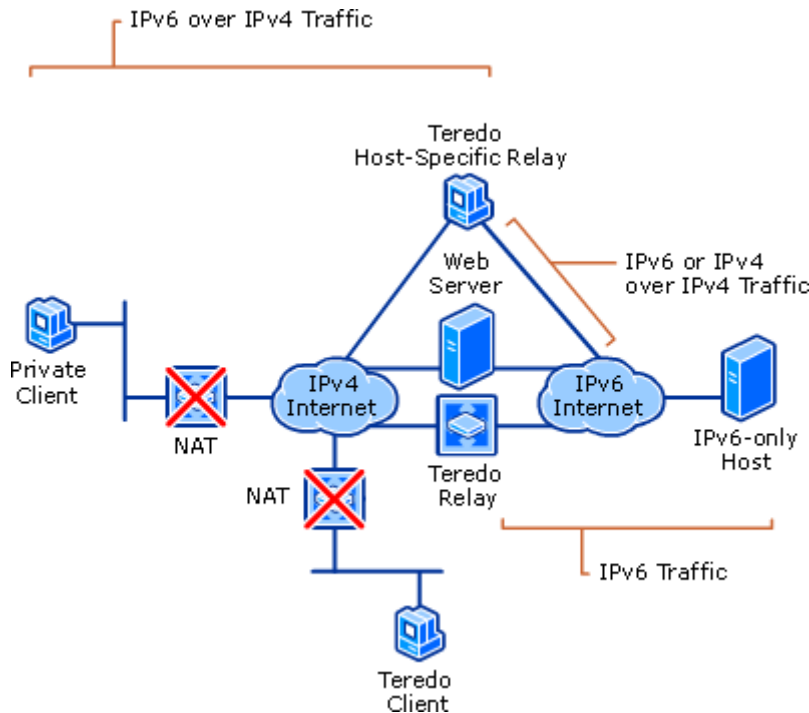
The following figure shows the components of the Teredo infrastructure:

# IPv6 Technical Reference

(Microsoft Corporation)

- Teredo clients
- Teredo servers
- Teredo relays
- Teredo host-specific relays

## Components of the Teredo Infrastructure



### Teredo Client

A Teredo client is an IPv6/IPv4 node that supports a Teredo tunneling interface through which packets are tunneled to either other Teredo clients or nodes on the IPv6 Internet (through a Teredo relay). A Teredo client communicates with a Teredo server to obtain an address prefix from which a Teredo-based IPv6 address is configured or to help initiate communication with other Teredo clients or hosts on the IPv6 Internet.

The Advanced Networking Pack for Windows XP includes a Teredo client.

### Teredo Server

A Teredo server is an IPv6/IPv4 node that is connected to both the IPv4 Internet and the IPv6 Internet, supports a Teredo tunneling interface over which packets are received. The general role of the Teredo server is to help configure addresses for Teredo clients and to facilitate the initial communication between Teredo clients and other Teredo clients or between Teredo clients and IPv6-only hosts. The Teredo server listens on UDP port 3544 for Teredo traffic.

The Advanced Networking Pack for Windows XP does not include Teredo server functionality. To facilitate communication between computers using the Advanced Networking Pack for Windows XP, Microsoft has deployed Teredo servers on the IPv4 Internet.

### Teredo Relay

## IPv6 Technical Reference (Microsoft Corporation)

A Teredo relay is an IPv6/IPv4 router that can forward packets between Teredo clients on the IPv4 Internet (using a Teredo tunneling interface) and IPv6-only hosts. In some cases, the Teredo relay interacts with a Teredo server to help it facilitate initial communication between Teredo clients and IPv6-only hosts. The Teredo relay listens on UDP port 3544 for Teredo traffic.

The Advanced Networking Pack for Windows XP does not include Teredo relay functionality. Microsoft does not plan to deploy any Teredo relays on the IPv4 Internet. Individual Internet service providers (ISPs) could deploy their own Teredo relays. The implementation of the Teredo client in the Advanced Networking Pack for Windows XP will work with a Teredo relay when sending traffic to an IPv6-only host on the IPv6 Internet. Teredo relays are not needed to communicate with Teredo host-specific relays.

### Teredo Host-specific Relay

Communication between Teredo clients and IPv6 hosts that are configured with global addresses must go through a Teredo relay. This is required for IPv6-only hosts connected to the IPv6 Internet. However, when the IPv6 host is enabled for both IPv6 and IPv4 and connected to both the IPv4 Internet and IPv6 Internet, then communication should occur between the Teredo client and the IPv6 host over the IPv4 Internet, rather than having to traverse the IPv6 Internet and go through a Teredo relay.

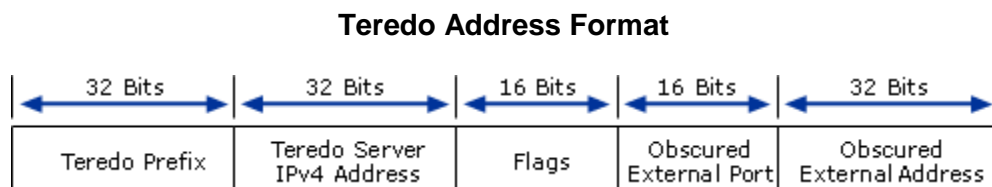
A Teredo host-specific relay is an IPv6/IPv4 node that has an interface and connectivity to both the IPv4 Internet and the IPv6 Internet and that can communicate directly with Teredo clients over the IPv4 Internet, without the need for an intermediate Teredo relay. The connectivity to the IPv4 Internet can be through a public IPv4 address or through a private IPv4 address and a neighboring NAT. The connectivity to the IPv6 Internet can be through a direct connection to the IPv6 Internet or through an IPv6 transition technology such as 6to4, where IPv6 packets are tunneled across the IPv4 Internet. The Teredo host-specific relay listens on UDP port 3544 for Teredo traffic.

The implementation of the Teredo client in the Advanced Networking Pack for Windows XP includes Teredo host-specific relay functionality. When the Advanced Networking Pack for Windows XP is installed, the Teredo host-specific relay functionality is automatically enabled if the computer running Windows XP SP1 has a global address assigned. A global address can be assigned from a Router Advertisement message from a native IPv6 router, an ISATAP router, or a 6to4 router. If the computer running Windows XP SP1 does not have a global address, Teredo client functionality is enabled.

The Teredo host-specific relay allows Teredo clients to efficiently communicate with 6to4 hosts, IPv6 hosts with a non-6to4 global prefix, or ISATAP or 6over4 hosts within organizations that use a global prefix for their addresses, provided both hosts are using the Advanced Networking Pack for Windows XP.

### Teredo Addresses

The following figure shows the format of Teredo addresses.



A Teredo address consists of the following:

## IPv6 Technical Reference (Microsoft Corporation)

- **Teredo prefix (Prefix):** The first 32 bits are for the Teredo prefix, which is the same for all Teredo addresses. The Internet Assigned Numbers Authority (IANA) has not yet defined this prefix, although the prefix 3FFE:831F::/32 is used for initial deployment.
- **Teredo server IPv4 Address:** The next 32 bits contain the IPv4 public address of the Teredo server that helped configure this Teredo address.
- **Flags:** The next 16 bits for are reserved for Teredo flags. The only defined flag is the high-order bit known as the Cone flag. The Cone flag is set when the NAT that is connected to the Internet is a cone NAT. The determination of whether the NAT that is connected to the Internet is a cone NAT occurs during the initial configuration of a Teredo client.
- **Obscured External Port:** The next 16 bits store an obscured version of the external UDP port that corresponds to all Teredo traffic for this Teredo client. When the Teredo client sends its initial packet to a Teredo server, the source UDP port of the packet is mapped by the NAT to a different, external UDP port. The Teredo client maintains this port mapping so that it remains in the NAT's translation table. Therefore, all Teredo traffic for the host uses the same external, mapped UDP port. The Teredo server determines the external UDP port from the source UDP port of the incoming initial packet that the Teredo client sent, and the Teredo server sends the port information back to the Teredo client. The external port is obscured by XORing the external port with 0xFFFF. For example, the obscured version of external port 5000 in hexadecimal format is EC77 (5000 = 0x1388, 0x1388 XOR 0xFFFF = 0xEC77). Obscuring the external port prevents NATs from translating the external port within the payload of the packets that they are forwarding.
- **Obscured External Address:** The last 32 bits store an obscured version of the external IPv4 address that corresponds to all Teredo traffic for this Teredo client. Just like the external port, when the Teredo client sends its initial packet to a Teredo server, the source IP address of the packet is mapped by the NAT to a different, external (public) address. The Teredo client maintains this address mapping so that it remains in the NAT's translation table. Therefore, all Teredo traffic for the host uses the same external, mapped, public IPv4 address. The Teredo server determines the external IPv4 address from the source IPv4 address of the incoming initial packet that the Teredo client sent, and the Teredo server sends the address information back to the Teredo client.

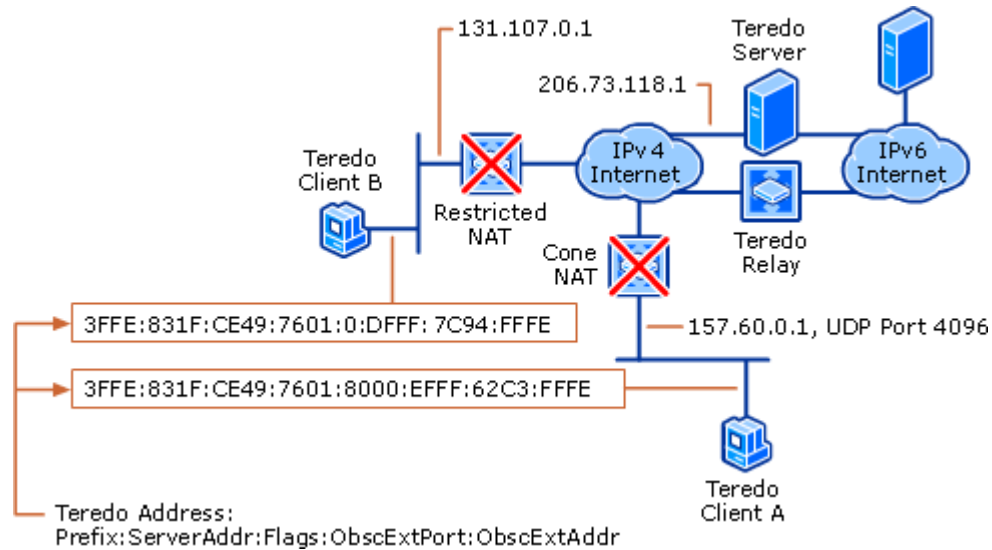
The external address is obscured by XORing the external address with 0xFFFFFFFF. For example, the obscured version of the public IPv4 address 131.107.0.1 in colon-hexadecimal format is 7C94:FFFE (131.107.0.1 = 0x836B0001, 0x836B0001 XOR 0xFFFFFFFF = 0x7C94FFFE). Obscuring the external address prevents NATs from translating the external address within the payload of the packets that they are forwarding.

The following figure shows an example of two Teredo clients and their addressing.

# IPv6 Technical Reference

(Microsoft Corporation)

## Teredo Addressing Example



For Teredo client A, the following are used to construct its Teredo address:

- Its external address and port for its Teredo traffic is 157.60.0.1, UDP port 4096.
- Its Teredo server is at the public IPv4 address of 206.73.118.1.
- It has determined that it is behind a cone NAT.

Therefore, using the Teredo address format of Prefix:ServerAddr:Flags:ObscExtPort:ObscExtAddr, the address for Teredo client A is 3FFE:831F:CE49:7601:8000:EFFF:62C3:FFFE. This address is based on the following:

- CE49:7601 is the colon-hexadecimal version of 206.73.118.1.
- 8000 is the Flags field in which the Cone flag is set to 1, indicating that Teredo client A's

NAT is a cone NAT.

- EFFF is the obscured version of 4096 (0x1000).
- 62C3:FFFE is the obscured version of 157.60.0.1.

For Teredo client B, the following are used to construct its Teredo address:

- Its external address and port for its Teredo traffic is 131.107.0.1, UDP port 8192.
- Its Teredo server is at the public IPv4 address of 206.73.118.1.
- It has determined that it is behind a restricted NAT.

Therefore, using the Teredo address format of Prefix:ServerAddr:Flags:ObscExtPort:ObscExtAddr, the address for Teredo client B is 3FFE:831F:CE49:7601:0:DFFF:7C94:FFFE. This address is based on the following:

- CE49:7601 is the colon-hexadecimal version of 206.73.118.1.
- 0 is the Flags field in which the Cone flag is set to 0, indicating that Teredo client B's NAT is a restricted NAT.

## IPv6 Technical Reference (Microsoft Corporation)

- DFFF is the obscured version of 8192 (0x2000).
- 7C94:FFFE is the obscured version of 131.107.0.1.

Teredo addresses are assigned to Teredo clients only. Teredo servers, relays, and host-specific relays are not assigned Teredo addresses.

### Communicating Using a Teredo Address

Initial configuration for Teredo clients is accomplished by sending a series of Router Solicitation messages to Teredo servers. The clients use the responses to derive a Teredo address and determine whether they are behind cone, restricted, or symmetric NATs. If a Teredo client is behind a symmetric NAT, then it cannot function. You can see what type of NAT a Teredo client has discovered from the display of the netsh interface ipv6 show teredo command.

Based on the Router Advertisement messages that the Teredo client receives, the Teredo client constructs its Teredo address from the following:

- The first 64 bits are set to the value that the Prefix Information option of the Router Advertisement message includes. The 64-bit prefix that the Teredo server advertises consists of the Teredo prefix (32 bits) and the public IPv4 address of the Teredo server (32 bits).
- The next 16 bits are either 0x8000 (cone NAT) or 0x0 (restricted NAT).
- The next 16 bits are set to the obscured external UDP port number that a special Teredo header in the Router Advertisement message includes.
- The last 32 bits are set to the obscured external IP address that a special Teredo header in the Router Advertisement message includes.

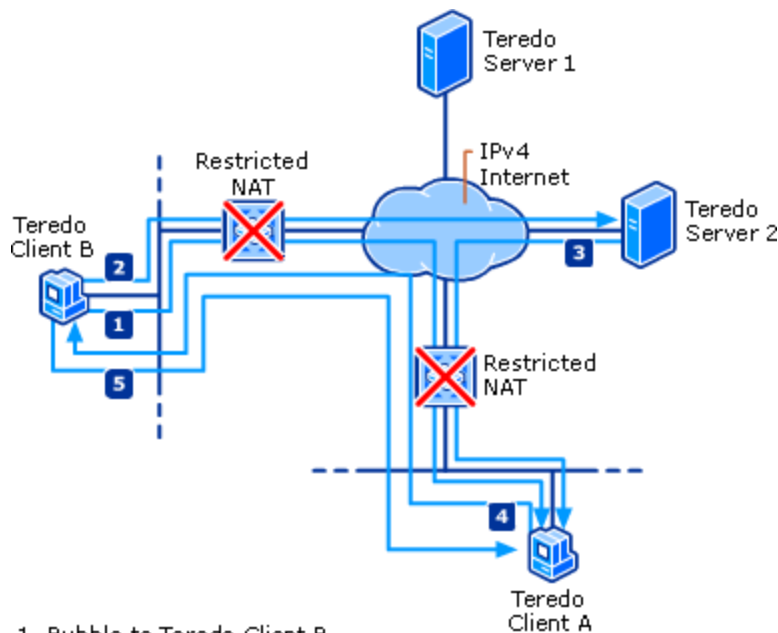
The success of initial communication between Teredo clients that are located in different sites depends on whether those sites are using cone NATs or restricted NATs.

When both Teredo clients are located behind cone NATs, the NAT translation table entries for Teredo traffic for each Teredo client allow traffic from any source IP address or source UDP port. Therefore, a Teredo client in one site can send packets directly to a Teredo client in another site without the use of additional packets to establish NAT translation table entries.

When both Teredo clients are located behind restricted NATs, the clients must establish additional NAT translation table entries before they can exchange unicast packets. The following figure shows the initial communication process between Teredo clients that are located in different sites when both sites are using restricted NATs.

## IPv6 Technical Reference (Microsoft Corporation)

### Initial Communication Process Between Teredo Clients that Are Located in Different Sites when Both Sites Are Using Restricted NATs



1. Bubble to Teredo Client B
2. Bubble to Teredo Server 2
3. Forwarded bubble to Teredo Client B
4. Bubble to Teredo Client A
5. Initial Packet to Teredo Client B

To send an initial communication packet from Teredo Client A to Teredo Client B, the following process is used:

1. Teredo Client A sends a bubble packet directly to Teredo Client B. A bubble packet contains no data and is used to create or maintain NAT mappings. Because Teredo Client B is behind a restricted NAT, Teredo traffic from an arbitrary source IPv4 address and UDP port number is not allowed unless the NAT translation table contains a source-specific entry. Assuming that no such entry exists, the restricted NAT silently discards the bubble packet. However, when the restricted NAT for Teredo Client A forwarded the bubble packet, it created a source-specific NAT translation table entry that will allow future packets sent from Teredo Client B to be forwarded to Teredo Client A.
2. Teredo Client A sends a bubble packet to Teredo Client B through Teredo Server 2 (Teredo Client B's Teredo server). The IPv4 destination address in the bubble packet is set to the IPv4 address of Teredo Server 2, which Teredo Client A determines from the third and fourth blocks of Teredo Client B's Teredo address.
3. Teredo Server 2 forwards the bubble packet to Teredo Client B. The restricted NAT for Teredo Client B forwards the packet because a source-specific mapping exists for Teredo traffic from Teredo Server 2 (established by the initial configuration of Teredo Client B).
4. Teredo Client B responds to the bubble packet received from Teredo Client A with its own bubble packet, which is sent directly to Teredo Client A. Because Teredo Client A's restricted NAT has a source-specific mapping for Teredo traffic from Teredo Client B (as established by the initial

## IPv6 Technical Reference (Microsoft Corporation)

bubble packet sent from Teredo Client A in step 1), it forwards the bubble packet to Teredo Client A.

5. When Teredo Client A receives the bubble packet from Teredo Client B, Teredo Client A determines that source-specific NAT mappings exist for both NATs. Teredo Client A sends an initial communication packet directly to Teredo Client B.

This process occurs transparently to the user at Teredo Client A.

### Related Information

For more information about TCP/IP, see "TCP/IP Technical Reference."

The standards for IPv6 are published in a series of documents called Requests for Comments (RFCs). RFCs are an evolving series of reports, proposals for protocols, and protocol standards that describe the internal workings of IPv6 and the Internet.

Although IPv6 standards are always published as RFCs, not all RFCs specify standards. RFCs are authored by individuals who voluntarily write and submit draft proposals for new protocols or specifications to the Internet Engineering Task Force (IETF) and other working groups. Submitted Internet drafts are first reviewed by a technical expert, a task force, or an RFC editor, and then the drafts are assigned a status.

If an Internet draft passes this initial review stage, the IETF circulates it to the larger Internet community for a period of time for further comment and review, and the IETF assigns an RFC number to the draft. This RFC number remains constant.

If the proposed specification changes, the IETF assigns a higher RFC number to the updated draft and circulates it. Higher RFC numbers identify more recent documents.

For more information, see the IETF RFC Database.

### Table of RFCs and Internet Drafts

The following table lists the RFCs and Internet drafts that relate to the implementation of the IPv6 protocol in Windows Server 2003 and Windows XP.

**Table of IPv6-Related RFCs and Internet Drafts**

RFC Number or Internet Draft	Title
1752	The recommendation for the IP next generation protocol
1828	IP authentication using keyed MD5
1886	DNS extensions to support IP version 6
1993	Transition mechanisms for IPv6 hosts and routers
1981	Path MTU discovery for IP version 6
2185	Routing aspects of IPv6 transition
2373	IP version 6 addressing architecture

## IPv6 Technical Reference (Microsoft Corporation)

2374	An IPv6 aggregatable global unicast address format
2401	Security architecture for the Internet protocol
2402	IP authentication header
2403	The use of HMAC-MD5-96 within ESP and AH (implemented for AH only)
2404	The use of HMAC-SHA-1-96 within ESP and AH (implemented for AH only)
2406	IP encapsulating security payload (ESP)
2428	FTP extensions for IPv6 and NATs
2460	Internet protocol, version 6 (IPv6) specification
2461	Neighbor discovery for IP version 6 (IPv6)
2462	IPv6 stateless address autoconfiguration
2463	Internet Control Message Protocol (ICMPv6) for the Internet protocol version 6 (IPv6) specification
2464	Transmission of IPv6 packets over Ethernet networks
2465	Management Information Base for IP Version 6: Textual Conventions and General Group (for Windows Server 2003 only)
2467	Transmission of IPv6 packets over FDDI networks
2526	Reserved IPv6 subnet anycast addresses
2529	Transmission of IPv6 over IPv4 domains without explicit tunnels
2553	Basic socket interface extensions for IPv6
2710	Multicast listener discovery (MLD) for IPv6 (implemented for host only)
2711	IPv6 router alert option (implemented for host only)
3041	Privacy extensions for stateless address autoconfiguration in IPv6
3056	Connection of IPv6 domains via IPv4 clouds
3484	Default Address Selection for Internet Protocol version 6 (IPv6)
Internet draft	An extension of format for IPv6 scoped addresses
Internet draft	Intra-site Automatic Tunnel Addressing Protocol (ISATAP)
3513	IP version 6 addressing architecture
Internet draft	IP version 6 scoped address architecture
Internet draft	Mobility support in IPv6 (host only)
Internet draft	Routing of scoped addresses in the Internet protocol version 6 (IPv6)
Internet draft	Site prefixes in neighbor discovery
Internet draft	The UDP lite protocol
Internet draft	IPv6 Near-Unique Site-Local Addresses
Internet draft	Management Information Base for the User Datagram Protocol (UDP)

## IPv6 Technical Reference (Microsoft Corporation)

Internet draft	Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6)
Internet draft	Default Router Preferences and More-Specific Routes

### IPv6 Tools and Settings

The following tools are associated with Internet Protocol version 6 (IPv6).

#### IPv6-aware TCP/IP Tools

Tool	Description
Ftp	Use to communicate with remote computers.
Ipconfig	Display current TCP/IP and IPv6 network configuration values, update or release Dynamic Host Configuration Protocol (DHCP) allocated leases, and display, register, or flush Domain Name System (DNS) names.
Ipsec6.exe	Use to configure Internet Protocol security (IPSec) policies and security associations between two IPv6 hosts.
Netsh	Provides 13 sets of commands for performing network configuration tasks. Windows Server 2003 adds a new context for managing IPv6 to the <b>netsh</b> command set.
Netstat	Display statistics for current TCP/IP connections. Windows Server 2003 adds IPv6 parameters to the <b>netstat</b> command.
Pathping	Trace a path to a remote system and report packet losses at each router along the way. Windows Server 2003 adds IPv6 parameters to the <b>pathping</b> command.
Ping	Send Internet Control Message Protocol (ICMP) Echo Requests to verify that TCP/IP is configured correctly and that a remote TCP/IP computer is available. Windows Server 2003 adds IPv6 parameters to the <b>ping</b> command.
Route	Display the IP routing table, and add or delete IPv6 routes.
Tracert	Trace a path to a remote system. Windows Server 2003 adds IPv6 parameters to the <b>tracert</b> command.

### FTP

The FTP connectivity tool included with Microsoft TCP/IP can be used to communicate with remote computers. The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003.

You can use the FTP tool to transfer files to and from a host running an FTP server service, such as the FTP component of Internet Information Services (IIS). The File Transfer Protocol (FTP) is a protocol that defines how to transfer files from one computer to another over a TCP/IP network, such as the Internet or a company intranet.

**Note:** The FTP service is a component of IIS. However, when you enable IIS on a server, FTP is not enabled unless you explicitly enable it. If you install IIS without FTP, you can use Add or Remove Windows Components in Add or Remove Programs in the Control Panel to install FTP later.

### Ipconfig

## IPv6 Technical Reference

(Microsoft Corporation)

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003.

You can use the Ipconfig command-line tool to display the current configuration of the installed IP stack on a networked computer and to refresh DHCP and DNS settings. The ipconfig command is often one of the first commands you use to check the status of the connection when you experience communication problems on a TCP/IP network. Ipconfig is most useful for managing computers that obtain an IP address automatically, such as by using DHCP or Automatic Private IP Addressing (APIPA).

When called with no parameters, Ipconfig displays the Internet Protocol version 4 (IPv4) address, subnet mask, and default gateway for all adapters on a computer. If the computer has an IPv6 address, Ipconfig also displays the IPv6 address information.

### **Ipsec6.exe**

The Ipsec6.exe tool is included on the Windows Server 2003 product CD. You can run Ipsec6.exe on computers running Windows Server 2003 and use Ipsec6.exe to target computers running versions of Windows that support IPv6 (Windows XP Service Pack 1 [SP1] and later and Windows Server 2003).

You can use Ipsec6.exe to configure IPSec policies and security associations between two IPv6 hosts. This configuration creates an IPSec security association (SA) between two hosts on the same subnet. The SA performs authentication by using the Authentication Header (AH) and either the Message Digest 5 (MD5) or Secure Hash Algorithm 1 (SHA1) hashed message authentication code (HMAC) algorithms. This configuration provides data origin authentication and data integrity for all traffic between two IPv6 hosts. Ipsec6.exe has multiple commands, each with its own set of parameters:

**ipsec6 sp [Interface]**

Displays the active security policies. Alternatively, displays the active security policies for a specific interface.

**ipsec6 sa**

Displays the active security associations.

**ipsec6 | FileNameWithNoExtension**

Loads the security policies from FileName.spd and the security associations from FileName.sad.

**ipsec6 s | FileNameWithNoExtension**

Saves the current security policies to FileName.spd and the current security associations to FileName.sad. You can use this command to create files that are used to configure security policy and security associations. When there are no security policies or security associations, this command creates FileName.spd for security policies and FileName.sad for security associations. You can use these files as templates to configure the desired security policies or security associations by modifying them with a text editor.

**ipsec6 d [{sp | sa}] [Index]**

## IPv6 Technical Reference (Microsoft Corporation)

Deletes the security policies (using the `sp` parameter) or security associations (using the `sa` parameter) from the list of active security policies and security associations, as specified by index number. You can use `ipsec6 sp` or `ipsec6 sa` to display the index number.

```
ipsec6 m [{on | off}]
```

Specifies whether binding updates that are used for mobile IPv6 are protected by IPsec. This is enabled by default.

**Note:** This implementation of IPsec for IPv6 is not recommended for use in a production environment because it relies on static keying and has no provisions for updating keys upon sequence number reuse. When you manually configure Security Parameters Indexes (SPIs), always use random numbers. Do not use sequential numbers for SPIs, or you will compromise the security of your IPsec for IPv6 policies. The IPv6 protocol for the Windows Server 2003 family supports the use of IPsec Encapsulating Security Payload (ESP) with NULL encryption.

### Netsh Commands for Interface IPv6

Netsh commands for Interface IPv6 are included on the Windows Server 2003 product CD. You can run Netsh commands for Interface IPv6 on computers running Windows Server 2003 and use Netsh commands for Interface IPv6 to target computers running versions of Windows that support IPv6 (Windows XP SP1 and later and Windows Server 2003).

The Netsh commands for Interface IPv6 provide a command-line tool that you can use to query and configure IPv6 interfaces, address, caches, and routes.

The Interface IPv6 context of netsh has a subcontext for 6to4, a transition technology described in the HowIPv6 Works section of this Technical Reference, that allows communication between IPv6/IPv4 nodes across the IPv4 Internet. You can use the commands in the netsh interface IPv6 6to4 context to configure or display the configuration of the IPv6 Helper service on either a 6to4 host or a 6to4 router. In addition, the Interface IPv6 context of netsh has a subcontext for Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). ISATAP is an address assignment and tunneling mechanism for communication between IPv6/IPv4 nodes within an IPv4 intranet. You can use the commands in the netsh interface ipv6 isatap context to configure the IPv4 address of the ISATAP router.

You can run these commands from the command prompt in the Windows Server 2003 family of products, or from the command prompt for the netsh interface IPv6 context. For these commands to work at the Windows Server 2003 family command prompt, you must type `netsh interface ipv6` before typing commands and parameters as they appear in the following reference.

#### 6to4

Specifies that the 6to4 context of netsh interface IPv6 6to4 is used.

#### Syntax

```
6to4
```

#### add6over4tunnel

Creates a 6over4 interface using the specified IPv4 address.

#### Syntax

```
add 6over4tunnel [[interface=]String] [localaddress=]IPv4Address  
[[store={active | persistent}]
```

# IPv6 Technical Reference

(Microsoft Corporation)

## Parameters

`[interface=]String`

Specifies an interface name or index.

`[localaddress=]IPv4Address`

Required. Specifies the IPv4 address that is encapsulated.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

This example command creates a 6over4 interface using the IPv4 address 10.1.1.1 on the interface named "Private."

```
add 6over4tunnel "Private" 10.1.1.1
```

## add address

Adds an IPv6 address to a specified interface. Time values can be expressed in days (d), hours (h), minutes (m), and seconds (s). For example, 2d represents two days.

## Syntax

```
add address [[interface=]String] [address=]IPv6Address [[type=]{unicast |  
anycast}] [[validlifetime=]{Integer | infinite}]  
[[preferredlifetime=]{Integer | infinite}] [[store=]{active | persistent}]
```

## Parameters

`[interface=]String`

Specifies an interface name or index.

`[address=]IPv6Address`

Required. Specifies the IPv6 address to add.

`[[type=]{unicast | anycast}]`

Specifies whether a unicast address (unicast) or an anycast address (anycast) is added. The default selection is unicast.

`[[validlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the address is valid. The default value is infinite.

`[[preferredlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the address is preferred. The default value is infinite.

# IPv6 Technical Reference

## (Microsoft Corporation)

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command adds the IPv6 address FE80::2 to the interface named "Private."

```
add address "Private" FE80::2
```

### add dns

Adds a new DNS server IP address to the statically-configured list of DNS servers for the specified interface.

### Syntax

```
add dns [interface=]String [address=]IPAddress [[index=]Integer]
```

### Parameters

```
[interface=]String
```

Required. Specifies, by name, which interface will have a DNS server IP address added to its list of DNS server IP addresses.

```
[address=]IPAddress
```

Required. Specifies the IPv6 address of the DNS server to add to the list.

```
[[index=]Integer]
```

Specifies the position on the statically-configured list in which to place the DNS server IP address specified in address. By default, the DNS server IP address is added to the end of the list.

### Remarks

If an index is specified, the DNS server is placed in that position in the list.

### Examples

In the first example command, a DNS server with the IPv6 address FEC0:0:0:FFFF::1 is added to the list of DNS server IP addresses for the interface named "Local Area Connection." In the second example, a DNS server with the IPv6 address FEC0:0:0:FFFF::2 is added at index 2 as the second server on the list of servers for the interface named "Local Area Connection."

```
add dns "Local Area Connection" FEC0:0:0:FFFF::1
add dns "Local Area Connection" FEC0:0:0:FFFF::2 index=2
```

### add prefixpolicy

Adds a source and destination address selection policy for a specified prefix.

### Syntax

```
add prefixpolicy [prefix=]IPv6Address/Integer [precedence=]Integer
[label=]Integer [[store=]{active | persistent}]
```

# IPv6 Technical Reference

(Microsoft Corporation)

## Parameters

`[prefix=]IPv6Address/Integer`

Required. Specifies the prefix for which to add a policy in the policy table. Integer specifies the prefix length.

`[precedence=]Integer`

Required. Specifies the precedence value used for sorting destination addresses in the policy table.

`[label=]Integer`

Required. Specifies the label value that allows for policies that require a specific source address prefix for use with a destination address prefix.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

This example command adds a prefix policy for prefix `::/96`, with a precedence value of 3 and a label value of 4.

```
add prefixpolicy ::/96 3 4
```

## add route

Adds a route for a specified prefix. Time values can be expressed in days (d), hours (h), minutes (m), and seconds (s). For example, 2d represents two days.

## Syntax

```
add route [prefix=]IPv6Address/Integer [[interface=]String]
[[nexthop=]IPv6Address] [[siteprefixlength=]Integer] [[metric=]Integer]
[[publish=]{no | yes | immortal}] [[validlifetime=]{Integer | infinite}]
[[preferredlifetime=]{Integer | infinite}] [[store=]{active | persistent}]
```

## Parameters

`[prefix=]IPv6Address/Integer`

Required. Specifies the prefix for which to add a route. Integer specifies the prefix length.

`[[interface=]String]`

Specifies an interface name or index.

`[[nexthop=]IPv6Address]`

Specifies the gateway address, if the prefix is not on-link.

`[[siteprefixlength=]Integer]`

## IPv6 Technical Reference

(Microsoft Corporation)

Specifies the prefix length for the entire site, if the prefix is not on-link.

```
[[metric=]Integer]
```

Specifies the route metric.

```
[[publish=]{no | yes | immortal}]
```

Specifies whether routes are advertised (yes), advertised with an infinite lifetime (immortal), or not advertised (no) in Route Advertisements. The default selection is no.

```
[[validlifetime=]{Integer | infinite}]
```

Specifies the lifetime over which the route is valid. The default value is infinite.

```
[[preferredlifetime=]{Integer | infinite}]
```

Specifies the lifetime over which the route is preferred. The default value is infinite.

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command adds a route on the interface named "Internet" with a prefix of 3FFE:: and a prefix length of 16 bits (3FFE::/16). The nexthop value is FE80::1.

```
add route 3FFE::/16 "Internet" FE80::1
```

### add v6v4tunnel

Creates an IPv6-in-IPv4 tunnel.

### Syntax

```
add v6v4tunnel [[interface=]String] [localaddress=]IPv4Address  
[remoteaddress=]IPv4Address [[neighbordiscovery=]{enabled | disabled}]  
[[store=]{active | persistent}]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[localaddress=]IPv4Address
```

Required. Specifies the IPv4 address of the local tunnel endpoint.

```
[remoteaddress=]IPv4Address
```

Required. Specifies the IPv4 address of the remote tunnel endpoint.

# IPv6 Technical Reference

## (Microsoft Corporation)

```
[[neighbordercovery=]{enabled | disabled}]
```

Specifies whether Neighbor Discovery is enabled (enabled) or disabled (disabled) on the interface. The default selection is disabled.

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command creates an IPv6-in-IPv4 tunnel between the local address 10.0.0.1 and the remote address 192.168.1.1 on the interface "Private."

```
add v6v4tunnel "Private" 10.0.0.1 192.168.1.1
```

### delete address

Modifies an IPv6 address on a specified interface.

### Syntax

```
delete address [[interface=]String] [address=]IPv6Address [[store=]{active  
| persistent}]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[address=]IPv6Address
```

Required. Specifies the IPv6 address to delete.

```
[[store=]{active | persistent}]
```

Specifies whether the deletion lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command deletes the address FE80::2 from the interface named "Private."

```
delete address "Private" FE80::2
```

### delete destinationcache

Clears the destination cache. If an interface is specified, clears the cache only on that interface. If an address is also specified, deletes only that destination cache entry.

### Syntax

```
delete destinationcache [[interface=]String] [[address=]IPv6Address]
```

### Parameters

# IPv6 Technical Reference

(Microsoft Corporation)

`[[interface=]String]`

Specifies an interface name or index.

`[[address=]IPv6Address]`

Specifies the address of the destination.

## Remarks

When no parameters are specified, all entries in the destination caches for all interfaces are deleted.

## Example

This example command deletes the destination cache for the interface named "Private."

```
delete destinationcache "Private"
```

## delete dns

Deletes statically-configured DNS server IPv6 addresses for a specific interface.

## Syntax

```
delete dns [[interface=]String [[address=]{IPv6Address | all}]
```

## Parameters

`[[interface=]String]`

Required. Specifies the interface, by name, for which you want to remove a DNS server from the list of DNS servers.

`[[address=]{IPv6Address | all}]`

Specifies the DNS server IPv6 address to delete. If all is specified, all DNS server IPv6 addresses on the list for the interface are deleted.

## Examples

In the first example command, the DNS server IPv6 address FEC0:0:0:FFFF::1 is deleted from the list of addresses for the connection named "Local Area Connection." In the second example command, all DNS server IPv6 addresses are deleted for the connection named "Local Area Connection."

```
delete dns "Local Area Connection" FEC0:0:0:FFFF::1
delete dns "Local Area Connection" all
```

## delete interface

Deletes a specified interface from the IPv6 stack.

## Syntax

```
delete interface [[interface=]String] [[store=]{active | persistent}]
```

## Parameters

`[[interface=]String]`

Specifies an interface name or index.

# IPv6 Technical Reference

(Microsoft Corporation)

```
[[store=]{active | persistent}]
```

Specifies whether the deletion lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Examples

This example command deletes the interface named "Private."

```
delete interface "Private"
```

## delete neighbors

Specifies that all entries in the neighbor cache are deleted. If an interface is specified, clears the cache only on that interface. If an address is also specified, deletes only that neighbor cache entry.

## Syntax

```
delete neighbors [[interface=]String] [[address=]IPv6Address]
```

## Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[[address=]IPv6Address]
```

Specifies the address of the neighbor.

## Example

This example command removes all entries from the neighbor cache on the interface named "Private."

```
delete neighbors "Private"
```

## delete prefixpolicy

Deletes the source and destination address selection policy for a specified prefix.

## Syntax

```
delete prefixpolicy [prefix=]IPv6Address/Integer [[store=]{active | persistent}]
```

## Parameters

```
[prefix=]IPv6Address/Integer
```

Required. Specifies the prefix (IPv6Address) and prefix length (Integer) to delete from the policy table.

```
[[store=]{active | persistent}]
```

Specifies whether the deletion lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

## IPv6 Technical Reference (Microsoft Corporation)

This example command deletes the prefix `::/96` from the policy table.

```
delete prefixpolicy ::/96
```

### delete route

Deletes an IPv6 route.

#### Syntax

```
delete route [prefix=]IPv6Address/Integer [[interface=]String]  
[[nexthop=]IPv6Address] [[store=]{active | persistent}]
```

#### Parameters

```
[prefix=]IPv6Address/Integer
```

Required. Specifies the prefix of the route to delete.

```
[[interface=]String]
```

Specifies an interface name or index.

```
[[nexthop=]IPv6Address]
```

Specifies the gateway address, if the prefix is not on-link.

```
[[store=]{active | persistent}]
```

Specifies whether the deletion lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

#### Example

This example command deletes the route with the prefix `3FFE::/16` and the gateway `FE80::1` from the interface named "Internet."

```
delete route 3FFE::/16 "Internet" FE80::1
```

### dump

Dumps the network adapter IPv6 configuration to the command prompt window when run within the netsh context. When used in a batch file or script, output can be saved in a text file.

#### Syntax

```
netsh interface ipv6 dump > [PathAndFileName]
```

#### Parameters

```
[PathAndFileName]
```

Specifies both the location to which to save the file and the name of the destination file to which the configuration is saved.

#### Remarks

## IPv6 Technical Reference

(Microsoft Corporation)

After file output is obtained, you can use the netsh exec command to either configure another computer with the same IPv6 configuration or to restore the original configuration on the same computer.

All IPv6 configuration information is saved with the dump command. For example, if an ISATAP or a 6to4 configuration is defined on an interface, the dump command saves these settings in the text file.

### Examples

In the first example, the command is run manually at the netsh interface ipv6 context of a command prompt. The IPv6 configuration is displayed in the command prompt window, and can be copied and pasted into a text file. In the second example, the dump command is run in a batch file, and the configuration is saved to a text file named ipv6\_conf.txt at the location C:\Temp.

### dump

```
netsh interface ipv6 dump > C:\temp\ipv6_conf.txt
```

### install

Installs IPv6.

### Syntax

```
install
```

### isatap

Specifies that the isatap context of netsh interface IPv6 isatap is used.

### Syntax

```
isatap
```

### Remarks

ISATAP is used for communication between IPv6 and IPv4 nodes within an IPv4 site. It is described in the Internet draft titled "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)."

### renew

Restarts IPv6 interfaces.

### Syntax

```
renew [[interface=]String]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

### Example

```
renew "Private"
```

### reset

Resets the IPv6 configuration state.

### Syntax

```
reset
```

# IPv6 Technical Reference

(Microsoft Corporation)

## set address

Modifies an IPv6 address on a specified interface. Time values can be expressed in days (d), hours (h), minutes (m), and seconds (s). For example, 2d represents two days.

### Syntax

```
set address [[interface=]String] [address=]IPv6Address [[type=]{unicast |  
anycast}] [[validlifetime=]{Integer | infinite}]  
[[preferredlifetime=]{Integer | infinite}] [[store=]{active | persistent}]
```

### Parameters

`[[interface=]String]`

Specifies an interface name or index.

`[address=]IPv6Address`

Required. Specifies the IPv6 address to modify.

`[[type=]{unicast | anycast}]`

Specifies whether the address is marked as a unicast address (unicast) or as an anycast address (anycast). The default selection is unicast.

`[[validlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the address is valid. The default value is infinite.

`[[preferredlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the address is preferred. The default value is infinite.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command sets the address FE80::2 on the interface named "Private" as an anycast address.

```
set address "Private" FE80::2 anycast
```

## set global

Modifies global configuration parameters.

### Syntax

```
set global [[defaultcurhoplimit=]Integer] [neighborcachelimit=]Integer  
[[routecachelimit=]Integer] [[reassemblylimit=]Integer] [[store=]{active |  
persistent}]
```

# IPv6 Technical Reference

(Microsoft Corporation)

## Parameters

`[[defaultcurhoplimit=]Integer]`

Specifies the default hop limit of packets sent.

`[[neighborcachelimit=]Integer]`

Required. Specifies the maximum number of neighbor cache entries.

`[[routecachelimit=]Integer]`

Specifies the maximum number of route cache entries.

`[[reassemblylimit=]Integer]`

Specifies the maximum size of the reassembly buffer.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

This example command sets global parameters for all IPv6-enabled interfaces on the computer. The default hop limit is set to 32, the maximum number of neighbor cache entries is set to 100, and the maximum number of route cache entries is 100,000.

```
set global 32 100 100000
```

## set interface

Modifies interface configuration parameters.

## Syntax

```
set interface [[interface=]String] [[forwarding=]{enabled | disabled}]  
[[advertise=]{enabled | disabled}] [[mtu=]Integer] [[siteid=]Integer]  
[[metric=]Integer] [[firewall=]{enabled | disabled}]  
[[siteprefixlength=]Integer] [[store=]{active | persistent}]
```

## Parameters

`[[interface=]String]`

Specifies an interface name or index.

`[[forwarding=]{enabled | disabled}]`

Specifies whether packets arriving on this interface can be forwarded to other interfaces. The default selection is disabled.

`[[advertise=]{enabled | disabled}]`

Specifies whether Router Advertisements are sent on this interface. The default selection is disabled.

# IPv6 Technical Reference

(Microsoft Corporation)

`[[mtu=]Integer]`

Specifies the MTU of this interface. The default MTU is the natural MTU of the link.

`[[siteid=]Integer]`

Specifies the site scope zone identifier.

`[[metric=]Integer]`

Specifies the interface metric, which is added to route metrics for all routes over the interface.

`[[firewall=]{enabled | disabled}]`

Specifies whether to operate in firewall mode.

`[[siteprefixlength=]Integer]`

Specifies the default length of the global prefix for the entire site.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

This example command sets the interface with the name "Private," with a siteid of 2 and a metric of 2. All other parameter values are left at the default values.

```
set interface "Private" siteid=2 metric=2
```

## set mobility

Modifies mobility configuration parameters.

## Syntax

```
set mobility [[security=]{enabled | disabled}]  
[[bindingcachelimit=]Integer] [[correspondentnode=]enabled | disabled]  
[[store=]{active | persistent}]
```

## Parameters

`[[security=]{enabled | disabled}]`

Specifies whether binding updates must be secured.

`[[bindingcachelimit=]Integer]`

Specifies the maximum number of binding cache entries.

`[[correspondentnode=]enabled | disabled]`

## IPv6 Technical Reference

(Microsoft Corporation)

Specifies whether Correspondent Node functionality is set to enabled or the default of disabled.

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

```
set mobility security=disabled bindingcachelimit=1000 corr=enabled
```

### set prefixpolicy

Modifies a source and destination address selection policy for a specified prefix.

### Syntax

```
set prefixpolicy [prefix=]IPv6Address/Integer [precedence=]Integer  
[label=]Integer [[store=]{active | persistent}]
```

### Parameters

```
[prefix=]IPv6Address/Integer
```

Required. Specifies the prefix for which to add a policy in the policy table. Integer specifies the prefix length.

```
[precedence=]Integer
```

Required. Specifies the precedence value used for sorting destination addresses in the policy table.

```
[label=]Integer
```

Required. Specifies the label value that allows for policies that require a specific source address prefix for use with a destination address prefix.

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

### Example

This example command sets a policy in the policy table for the prefix `::/96`, with a precedence value of 3 and a label value of 4.

```
set prefixpolicy ::/96 3 4
```

### set privacy

Modifies parameters related to temporary address generation. If `randomtime=` is specified, `maxrandomtime=` is not used. Time values can be expressed in days (d), hours (h), minutes (m), and seconds (s). For example, 2d represents two days.

### Syntax

```
set privacy [[state=]{enabled | disabled}] [[maxdadattempts=]Integer]  
[[maxvalidlifetime=]Integer] [[maxpreferredlifetime=]Integer]
```

# IPv6 Technical Reference

(Microsoft Corporation)

```
[[regeneratetime=]Integer] [[maxrandomtime=]Integer] [[randomtime=]Integer]  
[[store=]{active | persistent}]
```

## Parameters

```
[[state=]{enabled | disabled}]
```

Specifies whether temporary addresses are enabled.

```
[[maxdadattempts=]Integer]
```

Specifies the number of duplicate address detection attempts made. The default value is 5.

```
[[maxvalidlifetime=]Integer]
```

Specifies the maximum lifetime over which a temporary address is valid. The default value is 7d (seven days).

```
[[maxpreferredlifetime=]Integer]
```

Specifies the maximum lifetime over which an anonymous address is preferred. The default value is 1d (one day).

```
[[regeneratetime=]Integer]
```

Specifies the duration of time that elapses when a new address is generated prior to deprecating a temporary address. The default value is 5s (five seconds).

```
[[maxrandomtime=]Integer]
```

Specifies the upper limit to use when computing a random delay at boot. The default value is 10m (ten minutes).

```
[[randomtime=]Integer]
```

Specifies a time value to use, instead of a value generated at boot.

```
[[store=]{active | persistent}]
```

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## set route

Modifies route parameters. Time values can be expressed in days (d), hours (h), minutes (m), and seconds (s). For example, 2d represents two days.

## Syntax

```
set route [prefix=]IPv6Address/Integer [[interface=]String]  
[[nexthop=]IPv6Address] [[siteprefixlength=]Integer] [[metric=]Integer]  
[publish=]{no | yes | immortal} [[validlifetime=]{Integer | infinite}]  
[[preferredlifetime=]{Integer | infinite}] [[store=]{active | persistent}]
```

# IPv6 Technical Reference

(Microsoft Corporation)

## Parameters

`[[prefix=]IPv6Address/Integer]`

Required. Specifies the prefix (IPv6Address) and prefix length (Integer) of the route to modify.

`[[interface=]String]`

Specifies an interface name or index.

`[[nexthop=]IPv6Address]`

Specifies the gateway address, if the prefix is not on-link.

`[[siteprefixlength=]Integer]`

Specifies the prefix length for the entire site, if the prefix is not on-link.

`[[metric=]Integer]`

Specifies the route metric.

`[[publish=]{no | yes | immortal}]`

Specifies whether routes are advertised (yes), advertised with an infinite lifetime (immortal), or not advertised (no) in Route Advertisements. The default selection is no.

`[[validlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the route is valid. The default value is infinite.

`[[preferredlifetime=]{Integer | infinite}]`

Specifies the lifetime over which the route is preferred. The default value is infinite.

`[[store=]{active | persistent}]`

Specifies whether the change lasts only until the next boot (active) or is persistent (persistent). The default selection is persistent.

## Example

This example command sets a route on the interface named "Internet." The route prefix is 3FFE::, and has a length of 16 bits. The gateway address, defined by the nexthop parameter, is FE80::1.

```
set route 3FFE::/16 "Internet" FE80::1
```

## set state

Enables or disables IPv4 compatibility. The default value for all parameters is disabled.

## Syntax

```
set state [[v4compat=](enabled | disabled | default)]
```

# IPv6 Technical Reference

## (Microsoft Corporation)

### Parameters

```
[[v4compat=](enabled | disabled | default)]
```

Specifies whether IPv4 compatible interfaces are created. To both disable and delete IPv4 compatible interfaces, specify default. To disable IPv4 compatible interfaces without deleting them, specify disabled.

### Examples

In the first example command, IPv4-compatible addresses are disabled, and any previously existing interfaces are deleted. In the second example command, IPv4-compatible addresses are enabled.

```
set state default
set state v4compat=enabled
```

### show address

Displays all IPv6 addresses, or all addresses on a specified interface.

### Syntax

```
show address [[interface=]String] [[level=]{normal | verbose}]
[[store=]{active | persistent}]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[[level=]{normal | verbose}]
```

Specifies whether one line per interface is displayed (normal) or additional information is displayed for each interface (verbose). When no interface is specified, the default selection is normal. When an interface is specified, the default selection is verbose.

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) addresses are displayed. The default selection is active.

### show bindingcacheentries

Displays all binding cache entries.

### Syntax

```
show bindingcacheentries
```

### show destinationcache

Displays destination cache entries. If an interface is specified, displays the cache only on that interface. If an address is also specified, displays only that destination cache entry.

### Syntax

```
show destinationcache [[interface=]String] [[address=]IPv6Address]
```

### Parameters

## IPv6 Technical Reference

(Microsoft Corporation)

`[[interface=]String]`

Specifies an interface name or index.

`[[address=]IPv6Address]`

Specifies the address of the destination.

### **show dns**

Displays the DNS server configuration for a specific interface or interfaces.

### **Syntax**

`show dns [[interface=]String]`

### **Parameters**

`[[interface=]String]`

Specifies the interface, by name, for which you want to display configured DNS server IPv6 addresses. If no interface is specified, servers for all interfaces are displayed.

# IPv6 Technical Reference

## (Microsoft Corporation)

### Example

In this example command, DNS server IPv6 addresses configured on the “Local Area Connection” interface are displayed.

```
show dns "Local Area Connection"
```

### show global

Displays global configuration parameters.

### Syntax

```
show global [[store=]{active | persistent}]
```

### Parameters

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) information is displayed. The default selection is active.

### show interface

Displays information about all interfaces, or about a specified interface.

### Syntax

```
show interface [[interface=]String] [[level=]{normal | verbose}]  
[[store=]{active | persistent}]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[[level=]{normal | verbose}]
```

Specifies whether one line per interface is displayed (normal) or additional information is displayed for each interface (verbose). When no interface is specified, the default selection is normal. When an interface is specified, the default selection is verbose.

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) interfaces are displayed. The default selection is active.

### show joins

Displays all IPv6 multicast addresses, or all multicast addresses on a specified interface.

### Syntax

```
show joins [[interface=]String] [[level=]{normal | verbose}]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

# IPv6 Technical Reference

(Microsoft Corporation)

```
[[level=]{normal | verbose}]
```

Specifies whether one line per interface is displayed (normal) or additional information is displayed for each interface (verbose). When no interface is specified, the default selection is normal. When an interface is specified, the default selection is verbose.

## show mobility

Displays mobility configuration parameters.

### Syntax

```
show mobility [[store=]{active | persistent}]
```

### Parameters

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) information is displayed. The default selection is active.

## show neighbors

Displays neighbor cache entries. If an interface is specified, displays only the cache on that interface. If an address is also specified, displays only that neighbor cache entry.

### Syntax

```
show neighbors [[interface=]String] [[address=]IPv6Address]
```

### Parameters

```
[[interface=]String]
```

Specifies an interface name or index.

```
[[address=]IPv6Address]
```

Specifies the address of the neighbor.

## show prefixpolicy

Displays prefix policy table entries used in source and destination address selection.

### Syntax

```
show prefixpolicy [[store=]{active | persistent}]
```

### Parameters

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) information is displayed. The default selection is active.

## show privacy

Displays privacy configuration parameters.

### Syntax

# IPv6 Technical Reference

(Microsoft Corporation)

```
show privacy [[store=]{active | persistent}]
```

## Parameters

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) information is displayed. The default selection is active.

## show routes

Displays route table entries.

## Syntax

```
show routes [[level=]{normal | verbose}] [[store=]{active | persistent}]
```

## Parameters

```
[[level=]{normal | verbose}]
```

Specifies whether only normal routes (normal) or routes used for loopback (verbose) are displayed. The default selection is normal.

```
[[store=]{active | persistent}]
```

Specifies whether active (active) or persistent (persistent) routes are displayed. The default selection is active.

## show siteprefixes

Displays the site prefix table.

## Syntax

```
show siteprefixes
```

## uninstall

Uninstalls IPv6.

## Syntax

```
uninstall
```

## Netsh Interface IPv6 6to4

You can use the following commands in the netsh interface IPv6 6to4 context to display the configuration of or configure the 6to4 service on either a 6to4 host or a 6to4 router.

## set interface

Configures the 6to4 service on an interface.

## Syntax

```
set interface [name=] InterfaceName [[routing=] {enabled | disabled | default}]
```

## Parameters

```
[name=] InterfaceName
```

## IPv6 Technical Reference

(Microsoft Corporation)

Required. Specifies the name of the interface for which you want to set 6to4 service configuration. InterfaceName must match the name of the interface specified in Network Connections. If InterfaceName contains any spaces, it must be enclosed in quotes.

```
[[routing=] {enabled | disabled | default}]
```

Specifies whether the forwarding of 6to4 packets received on the interface is enabled, disabled, or set to its default value.

### Remarks

This command enables, disables, or sets to default the routing behavior of the 6to4 service on a specified interface.

The default setting for the routing parameter is enabled, which enables routing on private interfaces if Internet Connection Sharing (ICS) is used.

### show interface

Displays the 6to4 service routing configuration on all interfaces, or on a specified interface.

### Syntax

```
show interface [[name=] InterfaceName]
```

### Parameters

```
[[name=] InterfaceName]
```

Specifies the name of the interface for which you want to display the 6to4 service configuration. InterfaceName must match the name of the interface specified in Network Connections. If InterfaceName contains any spaces, it must be enclosed in quotes.

### Remarks

If an interface name is not specified, the 6to4 configuration for all interfaces is displayed.

### set relay

Configures the name of the 6to4 relay router for the 6to4 service. Additionally, specifies how often the name is resolved and the state of the relay component for the 6to4 service.

### Syntax

```
set relay [[name=] {RelayDNSName | default}] [[state=] {enabled | disabled  
| automatic | default}] [[interval=] {ResInterval | default}]
```

### Parameters

```
[[name=] {RelayDNSName | default}]
```

Specifies either the fully qualified domain name (FQDN) of a 6to4 relay router on the IPv4 Internet (RelayDNSName), or sets the relay name to its default value of 6to4.ipv6.microsoft.com (default).

```
[[state=] {enabled | disabled | automatic | default}]
```

Specifies whether the state of the relay component for the 6to4 service is enabled, disabled, automatically enabled if a public IPv4 address is configured, or set to its default value.

## IPv6 Technical Reference (Microsoft Corporation)

```
[[interval=] {ResInterval | default}]
```

Specifies how often the name of the relay router is resolved in minutes (ResInterval) or sets the resolution interval to its default value of 1440 minutes (default).

### Remarks

The 6to4 relay router is a router that provides an access point between the IPv4 Internet and the 6bone (the native IPv6 portion of the Internet). To access 6bone resources from a 6to4 router, the 6to4 router encapsulates 6to4 traffic with an IPv4 header and sends it to the IPv4 address of the 6to4 relay router. The 6to4 relay router removes the IPv4 header and forwards the traffic to the 6bone. For return traffic, the 6to4 relay router encapsulates IPv6 traffic and sends it to the 6to4 router at the 6to4 host's site.

The default name of the 6to4 relay router is 6to4.ipv6.microsoft.com. The default state is automatic, which enables the forwarding of native IPv6 traffic to a relay router when a public IPv4 address is assigned to any interface. The default resolution interval is 1440 minutes (once each day).

### show relay

Displays the relay router configuration for the 6to4 service.

### Syntax

```
show relay
```

### set routing

Sets both the state of routing and the inclusion of site-local address prefixes in Router Advertisements that are sent by the 6to4 router.

### Syntax

```
set routing [[routing=] {enabled | disabled | automatic | default}]  
[[sitelocals=] {enabled | disabled | default}]
```

### Parameters

```
[[routing=] {enabled | disabled | automatic | default}]
```

Specifies whether the state of routing on a 6to4 router is enabled, disabled, automatically enabled if ICS is enabled, or set to its default value.

```
[[sitelocals=] {enabled | disabled | default}]
```

Specifies whether the advertising of site-local address prefixes, in addition to 6to4 address prefixes, is enabled, disabled, or set to its default value.

### Remarks

The default setting for the routing parameter is automatic, which enables routing on private interfaces when ICS is used. The default setting for the sitelocals parameter is enabled, which enables the advertising of site-local prefixes when site-local addresses are configured on private interfaces.

### show routing

Displays the routing configuration of the 6to4 service.

### Syntax

# IPv6 Technical Reference

(Microsoft Corporation)

`show routing`

## set state

Configures the state of the 6to4 service.

### Syntax

```
set state [[state=] {enabled | disabled | default}] [[undoonstop=] {enabled | disabled | default}] [[6over4=] {enabled | disabled | default}]
```

### Parameters

```
[[state=] {enabled | disabled | default}]
```

Specifies whether the state of the 6to4 service is enabled, disabled, or set to its default value.

```
[[undoonstop=] {enabled | disabled | default}]
```

Specifies whether the reversal of all automatic configuration that has been performed by the 6to4 service occurs when the service stops is enabled, disabled, or set to its default value.

### Remarks

The default setting for the state parameter is enabled, which enables the 6to4 service. The default setting for the undoonstop parameter is enabled, which reverses all automatic configuration performed by the 6to4 service when the service is stopped.

## show state

Displays the state of the 6to4 service.

### Syntax

```
show state
```

## reset

Resets the 6to4 service.

### Syntax

```
reset
```

## Netsh Interface IPv6 ISATAP

You can use the following commands to configure the ISATAP router.

## set router

Specifies the ISATAP router information, including router name, state, and resolution interval.

### Syntax

```
set router [[name={String | default}] [[state={Enabled | Disabled | Default}] [[interval=Integer]
```

### Parameters

```
[[name={String | default}]
```

## IPv6 Technical Reference

(Microsoft Corporation)

Specifies whether the router is named with a string. If default is specified, the system reverts to using the default name.

```
[[state]={Enabled | Disabled | Default}]
```

Specifies whether the ISATAP router relays packets between subnets.

```
[[interval]=Integer]
```

Specifies the router resolution interval, in minutes. The default interval is 1440 (24 hours).

### Example

The following example command sets the router name to isatap, enables the router, and sets the resolution interval to 120 minutes:

```
set router isatap enabled 120
```

### show router

Displays configuration information for the ISATAP router.

### Syntax

```
show router
```

### Remarks

This command displays the router name, the relay state, and the resolution interval.

### Netstat

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003. You can use the Netstat command-line tool to display active TCP/IP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and User Datagram Protocol [UDP]), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).

### Pathping

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003. You can use Pathping, an IP packet route-tracing command-line tool that combines features of Ping and Tracert, to obtain additional information that neither of those tools provides. Specifically, you can use Pathping to discover the route to a remote host; it then pings the remote host for a period of time and collects and reports statistics. Pathping path information includes information about the intermediate routers visited on the path, the Round-Trip Time (RTT) value, and link-loss information.

### Ping

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003. You can use the Ping command-line tool as your primary tool for troubleshooting IP-level connectivity between two TCP/IP computers. Ping sends ICMP Echo Request or ICMPv6 messages to perform network diagnostics and to test reachability for a specific destination. By default, Ping queries for both IPv4 and IPv6 addresses and uses the addresses returned by the operating system. Ping lets you specify the size of

## IPv6 Technical Reference (Microsoft Corporation)

packets to use (the default is 32 bytes), how many to send, whether to record the route used, which Time-To-Live (TTL) value to use, and so on.

## IPv6 Technical Reference

(Microsoft Corporation)

### Route

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003. You can use the Route command-line tool to view and modify the local IP routing table. For two hosts to exchange IP datagrams, they must both have a route to each other, or they must use a default gateway that knows a route between the two. Typically, routers exchange information using a protocol such as Routing Information Protocol (RIP) or Open Shortest Path First (OSPF). RIP Listening service is available for Microsoft Windows XP Professional, and full routing protocols are supported by Windows Server 2003 in the Routing and Remote Access service.

### Tracert

The IPv6-aware version of this tool is included on the Windows Server 2003 product CD. You can run this command on computers running Windows XP or Windows Server 2003. You can use the Tracert command-line route-tracing tool to display the path between the sending host and a destination. The path that Tracert displays is a list of near-side router interfaces of the routers in the path between the source host and destination. Tracert uses the IP TTL field in ICMP Echo Requests and ICMP Time Exceeded messages to determine the path from a source to a destination through an IP internetwork.

Some routers silently drop packets with expired TTLs. These routers do not appear in the Tracert display.

Tracert works by incrementing the TTL value by one for each ICMP Echo Request it sends, and then waiting for an ICMP Time Exceeded message. The TTL values of the Tracert packets start with an initial value of one; the TTL of each trace after the first is incremented by one. A packet sent out by Tracert travels one hop further on each successive trip.

**Note:** The UNIX version of Tracert performs the same function as the Windows version, except that the IP payload is a UDP packet addressed to a (presumably) unknown destination UDP port. Intermediate routers send back ICMP Time Expired messages recording the route taken, and the final destination sends back an ICMP Destination Unreachable-Port Unreachable message. The UDP payload from the UNIX Tracert tool can cross routers and firewalls, whereas the ICMP Echo Request messages might not due to ICMP filtering. To avoid this problem in Windows Sever 2003, turn off packet filtering and then try using Tracert again.