

IP Terminology

Throughout this section, you'll learn several important terms vital to your understanding of the **Internet Protocol**. Here are a few to get you started:

- **Bit** - A bit is one binary digit, either a 1 or a 0.
- **Nibble** - A nibble is 4 bits.
- **Byte** - A byte is 8 bits.
- **Octet** - An octet, made up of 8 bits, is just an ordinary 8-bit binary number. In this section, the terms byte and octet are completely interchangeable.
- **Network Address** - This is the designation used in routing to send packets to a remote network—for example, **10.0.0.0**, **172.16.0.0**, and **192.168.10.0**.
- **IP Address** - A logical address used to define a single host; however, IP addresses can be used to reference many or all hosts as well. If you see something written as just IP, it is referring to IPv4. IPv6 will always be written as IPv6.
- **Broadcast Address** - The broadcast address is used by applications and hosts to send information to all hosts on a network. Examples include **255.255.255.255**, which designates all networks and all hosts; **172.16.255.255**, which specifies all subnets and hosts on network **172.16.0.0**; and **10.255.255.255**, which broadcasts to all subnets and hosts on network **10.0.0.0**.

The Hierarchical IP Addressing Scheme

An **IP address** consists of **32 bits** of information. These bits are divided into *four sections*, referred to as *octets* or *bytes*, and *four octets* sum up to *32 bits* (8 × 4 = 32). You can depict an IP address using one of three methods:

- Dotted-decimal, as in **172.16.30.56**
- Binary, as in **10101100.00010000.00011110.00111000**
- Hexadecimal, as in **AC.10.1E.38**

Each of these examples validly represents the same IP address. Hexadecimal is used with IPv6, and IPv4 addressing uses dotted-decimal or binary.

The 32-bit IP address is known as a *structured* or *hierarchical* address. The major advantage of the hierarchical scheme is that it can handle a large number of addresses, namely, 4.3 billion (a 32-bit address space gives you 2³², or 4,294,967,296). The disadvantage of the flat-addressing scheme and the reason it's not used for IP addressing relates to routing. If every address were unique, all routers on the Internet would need to store the address of each and every machine on the Internet. This would make efficient routing impossible, even if only a fraction of all possible addresses were used.

The solution to this problem is to use a two- or three-level hierarchical addressing scheme that is structured by **network and host** or by the **network, subnet, and host**.

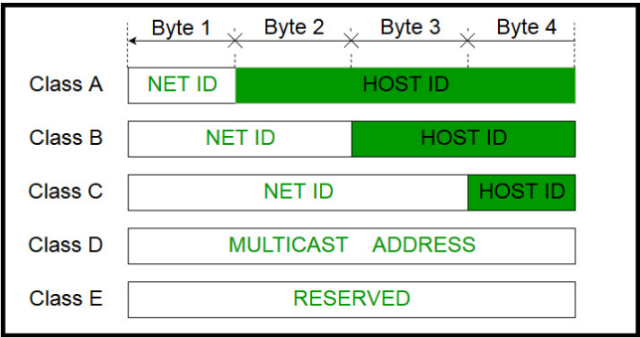
This two- or three-level scheme is comparable to a telephone number. The first section, the area code, designates a very large area. The second section, the prefix, narrows the scope to a local calling area. The final segment, the customer number, zooms in on the specific connection. IP addresses use the same type of layered structure. Rather than all 32 bits being treated as a unique identifier, as in flat addressing, a part of the address is designated as the network address and the other part is designated as either the subnet and host or just the host address.

Network Addressing

The **network address**—also called the **network number**—uniquely identifies each network. Every machine on the same network shares that network address as part of its IP address. In the IP address **172.16.30.56**, for example, **172.16** is the network address.

The host address is assigned to and uniquely identifies, each machine on a network. This part of the address must be unique because it identifies a particular machine as opposed to a network, which is a group. So in the sample IP address **172.16.30.56**, the **30.56** is the host address.

The designers of the Internet decided to create classes of networks based on network size. For the small number of networks possessing a very large number of hosts, they created the rank **Class A** network. At the other extreme is the **Class C** network, which is reserved for the numerous networks with a small number of hosts. The class distinction for networks between very large and very small is predictably the **Class B** network. Subdividing an IP address into a network and host address is determined by the class designation of your network. The below figure summarizes the classes of networks.



Network Classification

To ensure efficient routing, Internet designers defined a mandate for the leading-bits section of the address for each different network class. For example, since a router knows that a Class A network address always starts with a 0, the router might be able to speed a packet on its way after reading only the first bit of its address. This is where the address schemes define the difference between a Class A, a Class B, and a Class C address. The differences between these three classes will be explained in the following lessons.

Class A Addresses

In a **Class A network address**, the *first byte* is assigned to the *network address*, and the *three remaining bytes* are used for the *host addresses*. The Class A format is as follows:

network.host.host.host

For example, in the IP address **49.22.102.70**, the **49** is the network address and **22.102.70** is the host address. Every machine on this particular network would begin with the distinctive network address of **49**.

Class A network addresses are 1 byte long, with the first bit of that byte reserved and the 7 remaining bits available for manipulation, or addressing. As a result, the theoretical maximum number of Class A networks that can be created is 128.

The designers of the IP address scheme said that the first **bit** of the first **byte** in a Class A network address must always be off, or 0. This means a Class A address must be between 0 and 127 in the first byte, inclusive.

Consider the following network address:

0xxxxxxx

If we turn the other 7 bits all off and then turn them all on, we'll find the Class A range of network addresses:

00000000 = 0
01111111 = 127

So, a Class A network is defined in the first octet between 0 and 127, and it can't be less or more. To complicate matters further, the network address of all 0s (0000 0000) is reserved to designate the default route (see the below table, Reserved IP Addresses). Additionally, the address 127, which is reserved for diagnostics, can't be used either, which means that you can really only use the numbers 1 to 126 to designate Class A network addresses. This means the actual number of usable Class A network addresses is 128 minus 2, or 126.

Address	Function
Network address of all 0s	Interpreted to mean "this network or segment."
Network address of all 1s	Interpreted to mean "all networks."
Network 127.0.0.1	Reserved for loopback tests. Designates the local host and allows that host to send a test packet to itself without generating network traffic.
Host address of all 0s	Interpreted to mean "network address" or any host on the specified network.
Host address of all 1s	Interpreted to mean "all hosts" on the specified network; for example, 126.255.255.255 means "all hosts" on network 126 (Class A address).
Entire IP address set to all 0s	Used by Cisco routers to designate the default route. It could also mean "any network."
Entire IP address set to all 1s (same as 255.255.255.255)	Broadcast to all hosts on the current network; sometimes called an "all 1s broadcast" or limited broadcast.

Each *Class A address* has 3 bytes (24 bit positions) for the host address of a machine. This means there are 2²⁴—or 16,777,216—unique combinations and, therefore, precisely that many potential unique host addresses for each Class A network. Because host addresses with the two patterns of all 0s and all 1s are reserved, the actual maximum usable number of hosts for a Class A network is 2²⁴ minus 2, which equals 16,777,214.

Either way, you can see that’s a seriously huge number of hosts to have on a network segment! Here’s an example of how to figure out the valid host IDs in a Class A network address:

- All host bits off is the network address: 10.0.0.0.
- All host bits on is the broadcast address: 10.255.255.255.

The valid hosts are the numbers in between the network address and the broadcast address: 10.0.0.1 through 10.255.255.254. Notice that 0s and 255s can be valid host IDs. All you need to remember when trying to find valid host addresses is that the host bits can’t ever be all turned off or all turned on at the same time.

Class B Addresses

In a **Class B network address**, the first 2 bytes are assigned to the *network address* and the remaining 2 bytes are used for *host addresses*. The format is as follows:

network.network.host.host

For example, in the IP address 172.16.30.56, the network address is 172.16 and the host address is 30.56.

With a network address being 2 bytes (8 bits each), we’re left with 2¹⁶ unique combinations. But the Internet designers decided that all Class B network addresses should start with the binary digit 1, then 0. This leaves 14-bit positions available to manipulate, so in reality, we get 16,384 (that is, 2¹⁴) unique Class B network addresses.

In a Class B network, the RFCs state that the first bit of the first byte must always be turned on but the second bit must always be turned off. If we turn the other 6 bits all off and then all on, we will find the range for a Class B network:

10000000 = 128
10111111 = 191

As you can see, a Class B network is defined when the first byte is configured from 128 to 191.

A Class B address uses 2 bytes for host addresses. This is 2¹⁶ minus the two reserved patterns (all 0s and all 1s), for a total of 65,534 possible host addresses for each Class B network. Here’s an example of how to find the valid hosts in a Class B network:

- All **host** bits turned off is the network address: 172.16.0.0.
- All **host** bits turned on is the broadcast address: 172.16.255.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 172.16.0.1 through 172.16.255.254.

Class C Addresses

The first 3 bytes of a **Class C network address** are dedicated to the *network portion* of the address, with only 1 byte remaining for the *host address*. Here’s the format:

network.network.network.host

Using the example IP address 192.168.100.102, the network address is 192.168.100 and the host address is 102.

In a *Class C network address*, the first 3 bit positions are always the binary 110. The calculation is as follows: 3 bytes, or 24 bits, minus 3 reserved positions leave 21 positions. Hence, there are 2²¹, or 2,097,152, possible Class C networks.

For Class C networks, the RFCs define the first 2 bits of the first octet as always turned on, but the third bit can never be on. Following the same process as the previous classes, convert from binary to decimal to find the range. Here’s the range for a Class C network:

11000000 = 192
11011111 = 223

Class	Private IP address range	Subnet mask
A	10.0.0.0 – 10.255.255.255	255.0.0.0
B	172.16.0.0 – 172.16.31.255	255.255.0.0
C	192.168.0.0 – 192.168.255.255	255.255.255.0

So, if you see an IP address with a range from 192 up to 223, you’ll know it’s a Class C IP address.

Each unique Class C network has 1 byte to use for host addresses. This gets us to 2⁸, or 256, minus the two reserved patterns of all 0s and all 1s for a total of 254 available host addresses for each Class C network. Here’s an example of how to find a valid host ID in a Class C network:

- All host bits turned off is the network ID: 192.168.100.0.
- All host bits turned on is the broadcast address: 192.168.100.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 192.168.100.1 through 192.168.100.254.

Class D and E Addresses

Class D network addresses are not assigned to devices on a network. These addresses are used for *special-purpose, multicast applications* (such as video- and audio-streaming applications).

These addresses all need to be registered with **IANA** to be used globally. Addresses in this class have the first bits of the first octet set to 1110, yielding addresses in the first octet ranging from 11100000 to 11101111, or 224 to 239. These addresses are not defined by a normal subnet mask (we will discuss subnet mask later); instead, each address is used for a specific purpose. And because each address is individually used, it uses a 255.255.255.255 mask.

If Class D is special, **Class E addresses** are even more special. There is no defined use for this address class. Officially, it is listed as reserved for usage and testing by IANA and the Internet Research Task Force (IRTF). In fact, in 2002, Class E was updated to “reserved for future use.”

Class E comprises absolutely all valid addresses with 240 or higher in the first octet. The first bits of the first octet is 1111, which yields addresses from 11110000 to 11111111 — which, in decimals, are 240 to 255.

Because this address class is not being used for address allocation, you cannot know what the network ID, which defines the valid addresses in a range, is. So the inclusion of 255 at the end of the range is moot because this address range is not available for you to use. All you need to know is that by definition Class E includes all valid addresses higher than Class D.

Private IP Addresses (RFC 1918)

The people who created the IP addressing scheme also created what we call private IP addresses. These addresses can be used on a private network, but they’re not routable through the Internet. This is designed for the purpose of creating a measure of much-needed security, but it also conveniently saves valuable IP address space.

If every host on every network had to have real routable IP addresses, we would have run out of available IP addresses to hand out years ago. But by using private IP addresses, ISPs, corporations, and home users need only a relatively tiny group of bona fide IP addresses to connect their networks to the Internet. This is economical because they can use private IP addresses on their inside networks and get along just fine.

To accomplish this task, the ISP and the corporation—the end-users, no matter who they are—need to use something called **Network Address Translation (NAT)**, which basically takes a private IP address and converts it for use on the Internet. NAT provides security in that these IP addresses cannot be seen by external users. External users will only be able to see the public IP address to which the private IP address has been mapped. Moreover, multiple devices in the same private network can use the same, real IP address to transmit out onto the Internet. Doing things this way saves too much address space.

Address class	Reserved address space
Class A	10.0.0.0 through 10.255.255.255
Class B	172.16.0.0 through 172.31.255.255
Class C	192.168.0.0 through 192.168.255.255

Virtual IP

When a public IP address is substituted for the actual private IP address that has been assigned to the network interface of the device, the public IP address becomes an example of what is called a **virtual IP address**. This means it doesn’t correspond to an actual physical network interface.

There are other examples of such virtual IP addresses. For example, when a *web proxy server* substitutes its IP address for the sender’s IP address before sending a packet to the Internet, it is another example of creating a virtual IP address.

APIPA

What happens if you have a few hosts connected together with a switch or hub and you don't have a DHCP server? You can add static IP information to a host or you can let Windows provide what is called **Automatic Private IP Addressing (APIPA)**.

With **APIPA**, clients can automatically self-configure an IP address and subnet mask, which is the minimum information needed for hosts to communicate when a DHCP server isn't available. In this way, it could be thought of as a DHCP failover scheme. If all of the hosts set themselves with an APIPA address, they could communicate with one another but unfortunately not with any addresses that were statically configured, such as default gateways!

The IP address range for APIPA is **169.254.0.1** through **169.254.255.254**. The client also configures itself with a default Class B subnet mask of **255.255.0.0**. However, when you're in your corporate network and you're running a DHCP server and your host displays that it is using this IP address range, this means that either your DHCP client on the host is not working or the DHCP server is down or can't be reached because of a network issue. For example, if you plug a DHCP client into a port that is disabled, the host will receive an APIPA address.

Class	Private IP address range	Subnet mask
A	10.0.0.0 – 10.255.255.255	255.0.0.0
B	172.16.0.0 – 172.16.31.255	255.255.0.0
C	192.168.0.0 – 192.168.255.255	255.255.255.0

IPv4 Address Types

When a DHCP client broadcasted for an IP address; a router then forwarded this as a unicast packet to the DHCP server. With IPv4, broadcasts are pretty important, but with IPv6, there aren't any broadcasts sent at all.

Broadcast addresses were referred to in the earlier chapters. Here are the four IPv4 address types:

- **Layer 2 Broadcasts** - These are sent to all nodes on a LAN.
- **Broadcasts (Layer 3)** - These are sent to all nodes on the network.
- **Unicast** - This is an address for a single interface, and these are used to send packets to a single destination host.
- **Multicast** - These are packets sent from a single source and transmitted to many devices on different networks. Referred to as one-to-many.

Layer 2 Broadcasts

Layer 2 broadcasts are also known as hardware broadcasts—they only go out on a LAN, and they don't go past the LAN boundary (router). The typical hardware address (MAC address) is 6 bytes (48 bits) and looks something like **0c:a3:a4:f3:12:c2**. The broadcast would be all 1s in binary, which would be all Fs in hexadecimal, as in **ff:ff:ff:ff:ff:ff**.

Layer 3 Broadcasts

Broadcast messages are meant to reach all hosts on a broadcast domain. These are the network broadcasts that have all host bits on.

Here's an example: The network address of **172.16.0.0** would have a broadcast address of **172.16.255.255**—all host bits on. Broadcasts can also be "any network and all hosts," as indicated by **255.255.255.255**.

A good example of a broadcast message is an Address Resolution Protocol (ARP) request. When a host has a packet, it knows the logical address (IP) of the destination. To get the packet to the destination, the host needs to forward the packet to a default gateway if the destination resides on a different network. If the destination is on the local network, the source will forward the packet directly to the destination. Because the source doesn't have the MAC address to which it needs to forward the frame, it sends out a broadcast, something that every device in the local broadcast domain will listen to. This broadcast says, in essence, *"If you are the owner of IP address 192.168.2.3, please forward your MAC address to me"* with the source giving the appropriate information.

Unicast Address

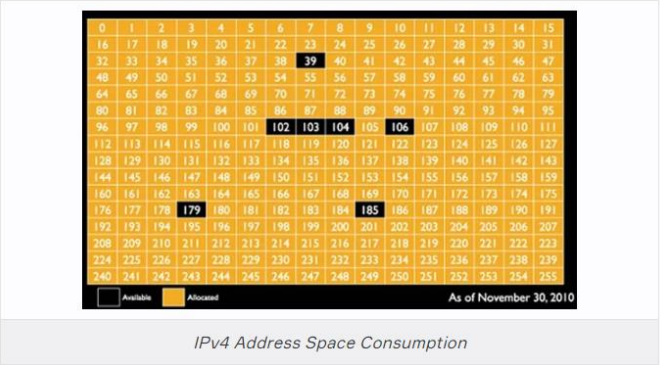
A unicast address is assigned to a single host, and this term is used in both IPv4 and IPv6 to describe your host interface IP address.

Multicast Address (Class D)

Multicast works by sending messages or data to multicast group addresses. Routers then forward copies (unlike broadcasts, which are not forwarded) of the packet out every interface that has hosts on a particular group address. This is where multicast differs from broadcast messages—with multicast communication, copies of packets, in theory, are sent only to subscribed hosts. This means that the hosts will receive, for example, a multicast packet destined for **224.0.0.10**. All hosts on the broadcast LAN will pick up the frame, read the destination address, and immediately discard the frame unless they are in the multicast group. This saves PC processing, not LAN bandwidth. Multicasting can cause severe LAN congestion, in some instances, if not implemented carefully.

There are several different groups that users or applications can subscribe to. The range of multicast addresses starts with **224.0.0.0** and goes through **239.255.255.255**. As you can see, this range of addresses falls within IP Class D address space based on a classful IP assignment.

Why Do We Need IPv6?



The above picture is old already but it shows you the reason why we need IPv6. We are running out of IPv4 addresses!

So what happened to IPv4? We have 32-bits which gives us 4,294,467,295 IP addresses. Remember our Class A, B, and C ranges. When the Internet started you would get a Class A, B, or C network. Class C gives you a block of 256 IP addresses, class B is 65,535 IP addresses, and class A even 16,777,216 IP addresses. Large companies like Apple, Microsoft, IBM, and such got one or more Class A networks. Did they really need more than 16 million IP addresses? Many IP addresses were just wasted.

We started using **VLSM (Variable Length Subnet Mask)** so we could use any subnet mask we like and create smaller subnets, we had to use the class A, B or C networks for a while. We also started using **NAT** and **PAT** so we can have many private IP addresses behind a single public IP address. (Subnets, NAT, and PAT will be discussed later.)

Nevertheless, the Internet has grown in a way nobody expected 20 years ago. Despite all tricks like *VLSM* and *NAT/PAT* we really need more IP addresses and that's why we need IPv6.

So what happened to IPv5? IP version 5 was used for an experimental project called *"Internet Stream Protocol"*.

Q: What do you mean by IPv6?

A: IPv6 stands for Internet Protocol version 6 and is the latest version of the Internet Protocol. The IP address length is 128 bits which resolves the issue of approaching shortage of network addresses.

- Interview Q&A

Internet Protocol Version 6 (IPv6)
The Benefits of and Uses for IPv6

IPv6 has 128-bit addresses and has a much larger address space than 32-bit IPv4 which offered us a bit more than 4 billion addresses. Keep in mind every additional bit doubles the number of IP addresses...so we go from 4 billion to 8 billion, 16, 32, 64, etc. Keep doubling until you reach 128 bit. With 128 bits this is the largest value you can create:

340,282,366,920,938,463,463,374,607,431,768,211,456

This gives us enough IP addresses for networks on earth, the Moon, Mars, and the rest of the universe. To put this in perspective let's put the entire IPv6 and IPv4 address space next to each other:

IPv6: 340282366920938463463374607431768211456
IPv4: 4294467295

[illegible]

The main reason to start using IPv6 is that we need more addresses but it also has some benefits:

- **More Efficient Routing** – IPv6 reduces the size of routing tables and makes routing more efficient and hierarchical. In IPv6 networks, fragmentation is handled by the source device, rather than a router.
- **More efficient packet processing** – Compared with the IPv4, IPv6 contains no IP-level checksum, so the checksum does not need to be recalculated at every router hop.
- **Directed Data Flows** – IPv6 supports multicast rather than broadcast. Multicast allows bandwidth-intensive packet flows to be sent to multiple destinations simultaneously, saving network bandwidth.
- **Simplified network configuration** – IPv6 devices can independently auto-configure themselves when connected to other IPv6 devices. Configuration tasks that can be carried out automatically include IP address assignment and device numbering.
- **Security** – IPSec security, which provides confidentiality, authentication, and data integrity, is engraved into IPv6.

IPv6 Addressing and Expressions

What does an IPv6 address look like? We use a different format than IPv4. We don't use decimal numbers like for IPv4, we are using hexadecimal now. Here's an example of an actual IPv6 address:

2041:1234:140F:1122:AB91:564F:875B:131B

As you can now see, the address is truly much larger and it has eight groups of numbers instead of four, also that those groups are separated by colons instead of periods. One other thing that should be pointed out is for when you set up your test network with IPv6, you have to type the address into the browser with brackets around the literal address. Because a colon is already being used by the browser for specifying a port number. So basically, if you don't enclose the address in brackets, the browser will have no way to identify the information. Here's an example of how this looks:

http://[2001:0db8:3c4d:0012:0000:0000:1234:56ab]/default.html

Shortening IPv6 Addresses

IPv6 addresses are hexadecimal and since they are 128-bit, they are quite long. Imagine you have to call a friend and ask him/her to ping the following address:

2041:0000:140F:0000:0000:875B:131B

To make our lives a bit better, IPv6 addresses can be shortened. Let's take a look at some examples and see how it works:

```
Original: 2041:0000:140F:0000:0000:875B:131B
Short: 2041:0000:140F::875B:131B
```


If there is a string of zeros then you can remove them once. In the example above the entire 0000:0000:0000 part was removed. You can only do this once so you cannot do this:

```
Original: 2001:0000:0000:0012:0000:0000:1234:56ab
Wrong!: 2001::12::1234:56ab
```

There is more however, the address can be shortened even more:

```
Short: 2041:0000:140F::875B:131B
Shorter: 2041:0:140F::875B:131B
```

If you have a "hextet" with 4 zeros then you can remove those and leave a single zero. Your IPv6 device will add the remaining 3 zeros.

 **Tip:**

- When we talk about IPv4 addresses, we use the term “octet” to define a “block” of 8 bits. In IPv6, there is no official term (yet) and there is an IETF draft that discusses the names to be used. The official term for 4 hexadecimal values is “hexadectet”, this is hard to remember/pronounce so the short form “hextet” will be used.

Leading zeros can also be removed, here's another address to demonstrate this:

```
Original: 2001:0001:0002:0003:0004:0005:0006:0007
Short: 2001:1:2:3:4:5:6:7
```

By removing these zeros we get a nice short IPv6 address.

To summarize these rules:

- An entire string of zeros can be removed, you can only do this once.
- 4 zeros can be removed, leaving only a single zero.
- Leading zeros can be removed.

Address Types

IPv4 uses **unicast**, **broadcast**, and **multicast addresses**, which basically define how many other devices we're communicating to. IPv6 introduces the **anycast address** type. Broadcasts, as we know them, have been eliminated in IPv6 because of their cumbersome inefficiency.

- Unicast Address

Unicast addresses are same as in IPv4, packets are delivered to a single node.
Unicast addresses can be divided into two categories:

- ## 1. Global Unicast Addresses

These are worldwide unique addresses that a network device needs in order to connect to the internet. The format prefix is usually 2000::/3 and includes all addresses that begin with 2000 to 3FFF. These types of addresses are routable and can be used to directly address a host in the local network over the internet. Global unicast addresses that are distributed to end-users begin with the hexadecimal block 2001.

- ## 2. Link-Local Addresses

Addresses in this category are only valid within local networks and begin with the format prefix FE80::10. Local link addresses are used to address elements within a local network and are used, for example, for auto-configuration.

- Multicast

As in IPv4, packets addressed to a multicast address are delivered to all hosts identified by the multicast address. Sometimes it is called one-to-many addresses. It's really easy to spot multicast addresses in IPv6 because they always start with FF.

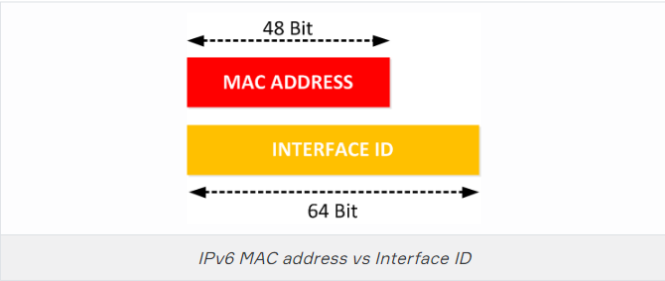
Special Addresses

The below table lists some of the addresses and address ranges that are all special or reserved for a specific use.

Address	Meaning
0:0:0:0:0:0:0:0	Equals ::. This is the equivalent of IPv4's 0.0.0.0 and is typically the source address of a host before the host receives an IP address when you're using DHCP-driven stateful configuration.
0:0:0:0:0:0:0:1	Equals ::1. The equivalent of 127.0.0.1 in IPv4. 0::FFFF:192.168.100.1 This is how an IPv4 address would be written in a mixed IPv6/IPv4 network environment.
2000::/3	The global unicast address range allocated for Internet access.
FE80::/10	The link-local unicast range.
FF00::/8	The multicast range.
3FFF:FFF::/32	Reserved for examples and documentation.
2001:0DB8::/32	Also reserved for examples and documentation.
2002::/16	Used with 6to4 tunneling, which is an IPv4-to-IPv6 transition system.

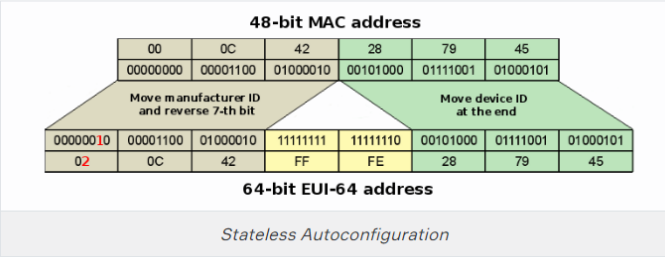
Stateless Autoconfiguration (EUI-64)

EUI-64 (Extended Unique Identifier) is a method we can use to automatically configure IPv6 host addresses. An IPv6 device will use the MAC address of its host to generate a unique 64-bit interface ID. However, a MAC address is 48 bit and the host ID is 64 bit. What are we going to do with the missing bits?



Here's what we will do to fill the missing bits:

- 1. We take the MAC address and split it into two pieces.
- 2. We insert "FFFE" in between the two pieces so that we have a 64-bit value.
- 3. We invert the 7th bit of the interface ID. So if the MAC address would be 00:0c:42:28:79:45 then this is what the interface ID will become:



Here are a few examples:

- MAC address 0090:2716:fd0f
- IPv6 EUI-64 address: 2001:0db8:0:1:0290:27ff:fe16:fd0f

That one was easy! So let's do another:

- MAC address aa12:bcbc:1234
- IPv6 EUI-64 address: 2001:0db8:0:1:a812:bfff:febc:1234

10101010 represents the first 8 bits of the MAC address (aa), which when inverting the 7th bit becomes 10101000. The answer becomes a8. This is important for you to understand.

- MAC address 0c0c:dede:1234
- IPv6 EUI-64 address: 2001:0db8:0:1:0e0c:deff:fede:1234

0c is 00001100 in the first 8 bits of the MAC address, which then becomes 00001110 when flipping the 7th bit. The answer is then 0e. Let's practice one more:

- MAC address 0b34:ba12:1234
- IPv6 EUI-64 address: 2001:0db8:0:1:0934:baff:fe12:1234

0b in binary is 00001011, the first 8 bits of the MAC address, which then becomes 00001001. The answer is 09.

Migrating to IPv6

- Dual Stacking

This is the most common type of migration strategy because, well, it's the easiest on us—it allows our devices to communicate using either IPv4 or IPv6. Dual stacking lets you upgrade your devices and applications on the network one at a time. As more and more hosts and devices on the network are upgraded, more of your communication will happen over IPv6, and after you've arrived—everything's running on IPv6 and you get to remove all the old IPv4 protocol stacks you no longer need.

- 6to4 Tunneling

6to4 tunneling is really useful for carrying IPv6 packets over a network that's still running IPv4. It's quite possible that you'll have IPv6 subnets or other portions of your network that are all IPv6, and those networks will have to communicate with each other. The whole idea of tunneling isn't a difficult concept, and creating tunnels really isn't as hard as you might think. All it really comes down to is snatching the IPv6 packet that's traveling across the network and sticking an IPv4 header onto the front of it.

