**Chapter 26: IPv6 Addressing**

Table of Contents

[Colon Hexadecimal Notation 3](#_Toc86834227)

[Zero Compression 3](#_Toc86834228)

[CIDR Address 4](#_Toc86834229)

[26.2 Address Space Allocation 5](#_Toc86834230)

[Prefixes 7](#_Toc86834231)

[Unspecified Address 8](#_Toc86834232)

[Loopback Address 8](#_Toc86834233)

[Embedded Addresses 8](#_Toc86834234)

[Unique Local Unicast Address 9](#_Toc86834235)

[Link Local Address 10](#_Toc86834236)

[Multicast Address 11](#_Toc86834237)

[26.3 Global Unicast Addresses 12](#_Toc86834238)

[Interface Identifier 12](#_Toc86834239)

[Mapping EUI-64 Addresses 13](#_Toc86834240)

[Mapping Ethernet MAC Addresses 13](#_Toc86834241)

[26.4 Autoconfiguration 15](#_Toc86834242)

**IPv4** uses 32-bit addresses, which causes a problem. Basically, we are running out of addresses and are having to come up with increasingly creative ways to distribute the addresses that we do have. This is why we have techniques like subnetting or the use of NAT routers.

IPv6 on the other hand, provides **16 byte** or **128-bit** addresses. This won’t get exhausted anytime soon. Of this, **64 bits** are used as the **Network Identifier** and **64 bits** are used as the **Interface Identifier** by default. Note that this fixed definition only applies for unicast and anycast addresses, not multicast ones.

## Colon Hexadecimal Notation

In IPv4, we used **Dotted Decimal Notation**, where each octet was converted to a decimal number, with the four octets separated by dots. If we use this with IPv6, we will end up with 16 octets, which is hard to manage.

Instead, we use the **Colon Hexadecimal Notation**, where the 128 bits are divided into **8 sections**, each of **16 bits**, and each section’s value is represented as its **hexadecimal** equivalent. The sections are separated with **colons**.

### Zero Compression

Most of time, we get s in most of the octets. To make things easier for us, we can compress the address. This process is called **zero compression**.

Note that zero compression can only be done once per address. The above address had two sections of s, but only one was compressed. This is because we would be unable to tell how many s were compressed from the first section and how many from the second section.

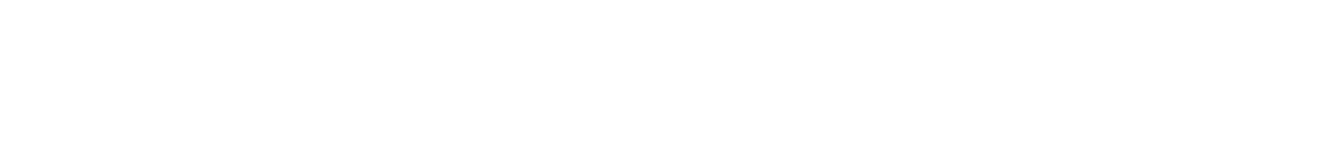
There was another type of compression here, where the was converted to .

### CIDR Address

We can also use **CIDR** notation with IPv6 addresses to denote how many bits are being used for the network portion.

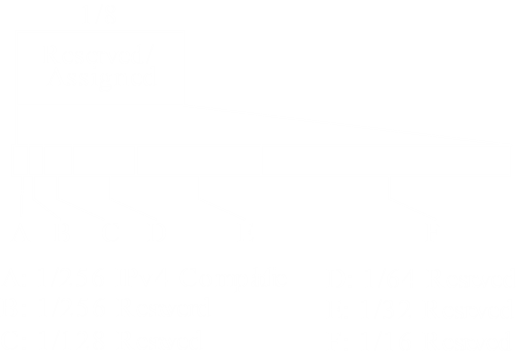
## 26.2 Address Space Allocation

The total address space available in IPv6 consists of  **addresses**. This address space can be divided into 8 portions:



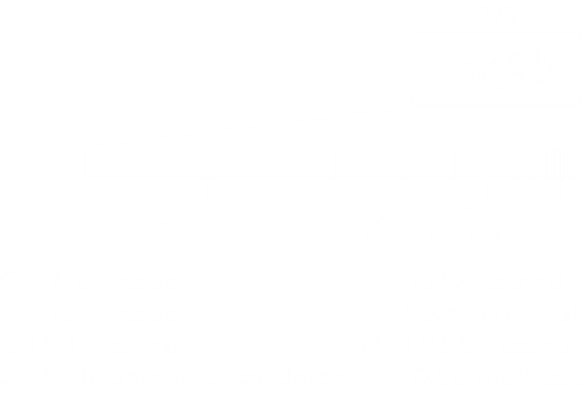
As can be see, 5/8th of the total space is completely reserved and not being used for anything. Only 1/8th is used for **Global Unicast** addresses, which is just normal usage. The first and last 1/8th of the address space are **Reserved/Assigned**. Essentially, those addresses have special uses.

The first Reserved/Assigned space is further divided into **6 parts**. Here, the first part, **Part A**, consists of 1/256th of the total address space. It is the only portion that is used. All the others are reserved.



Part A is used for **IPv4 compatibility**. Basically, this section consists of IPv6 addresses that are IPv4 compatible. How this works will be covered later. If we face a scenario where the source uses IPv6 but the destination uses IPv4 or some router in between uses IPv4, then we use one of the addresses from this portion. Such addresses are also called **embedded** addresses, since an IPv4 address is embedded inside an IPv6 address.

The last Reserved/Assigned space is divided into 8 portions. Even here, only the parts J, L and N are used while the other five are reserved.



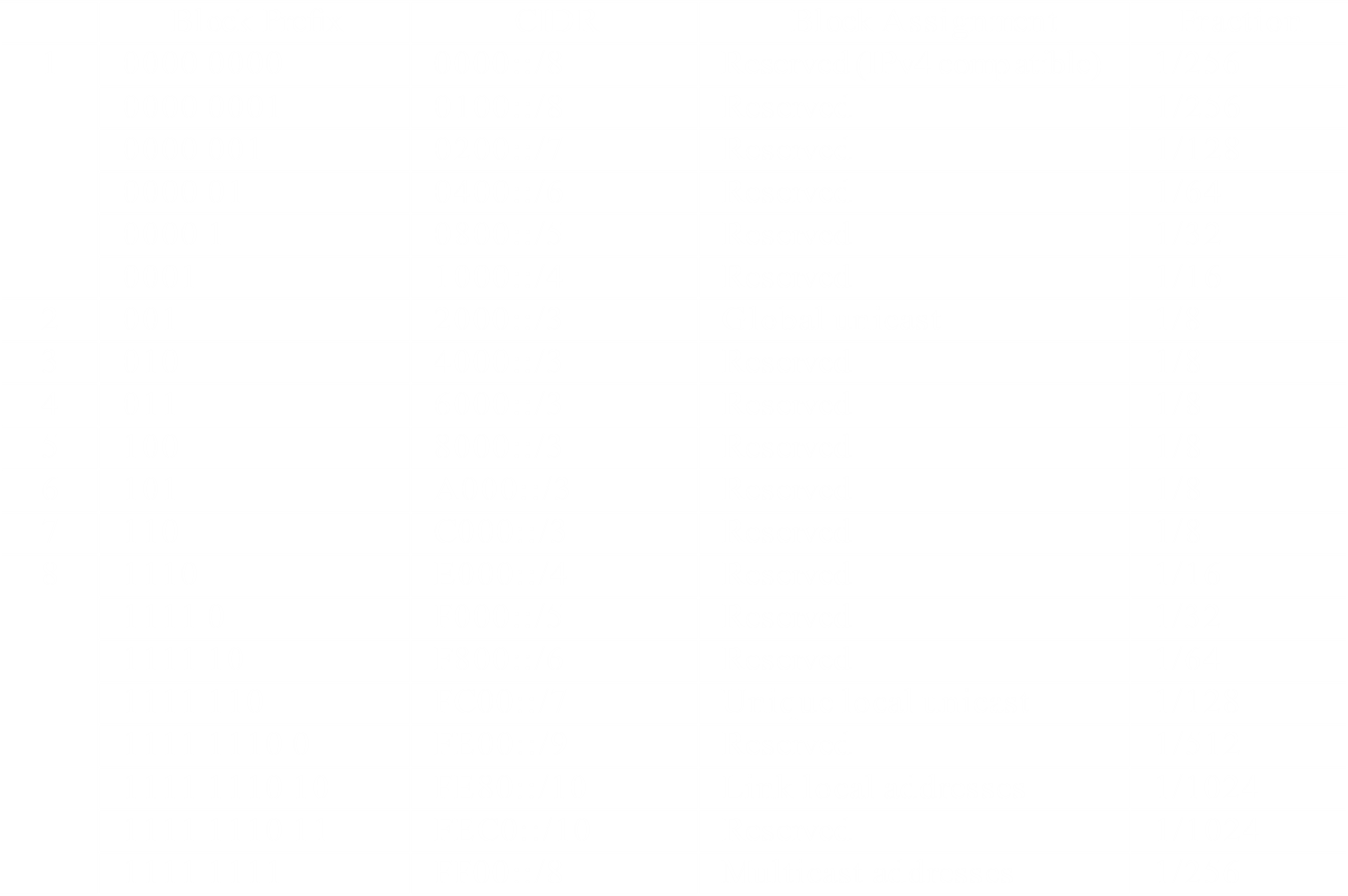
Part J consists of 1/128th of the total address space. It contains **Unique Local Unicast** addresses. In IPv4, we had **private address**, which were used to allow subdivisions inside a single network. However, in IPv6, we do not need to do this, since there are an abundance of addresses. We still need private addresses though, since some may wish to have addresses that are not routable via the internet.

Part L consists of 1/1024th of the total address space. It contains **Link Local** addresses. These are also private addresses. The difference is that these are at **link-level**, while those in Part J are at **site-level**. Site-level addresses are for large organizations. Again, we will be looking into the details of this later.

Part N consists of 1/256th of the total address space. It contains **multicast** addresses. IPv6 does not use broadcast addresses. Instead, it uses multicast addresses for broadcast purposes.

### Prefixes

Each of the sections of the address space can be identified using some **prefixes**.



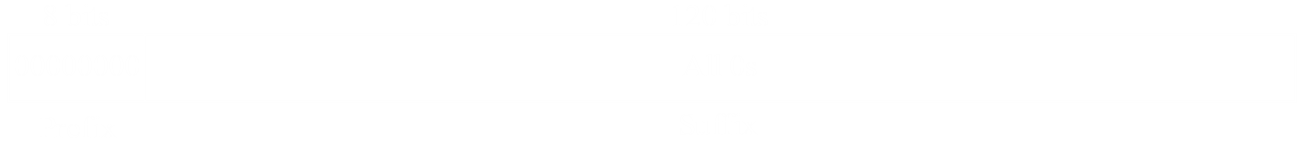
In the above table, block prefixes actually consist of the first 16 bits, which is how we are getting the CIDR notation. However, only the first few bits of the 16 bits is required to identify the block.

For **Global Unicast** addresses, just the first 3 bits are reserved. This leaves us with addresses.

### Unspecified Address

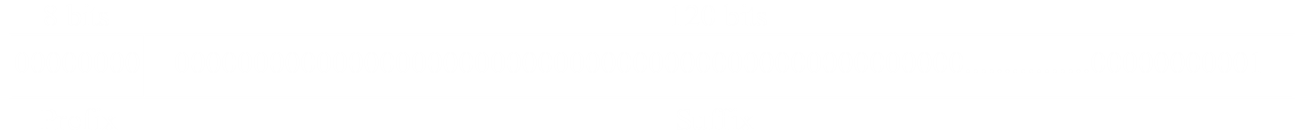
In IPv4, we had a special address called an **All-Zeroes Address**. This was used during the bootstrap process, when the host did not know its own address, so it uses this address as the source address.

In IPv6, we have a similar address called an **Unspecified Address**, . This is **compatible with IPv4**.



### Loopback Address

A **Loopback Address** is used by a host to send a packet to itself, for testing purposes. In IPv6, this address is , meaning all s and the last bit set to . This is **compatible with IPv4**.



Notice that we only have a **single address** as a loopback address in IPv6, whereas we had an entire block of addresses, , in IPv4.

### Embedded Addresses

**Embedded Addresses** are used when we need an IPv6 address to be IPv4 compatible. They can be of two types, **Compatible Addresses** and **Mapped Addresses**.

A **Compatible Address** contains all s and just has the **last 32 bits** as an IPv4 address.



Because of this, IPv4 compatible addresses use **mixed representation**. For example, is an example of an IPv4 compatible address.

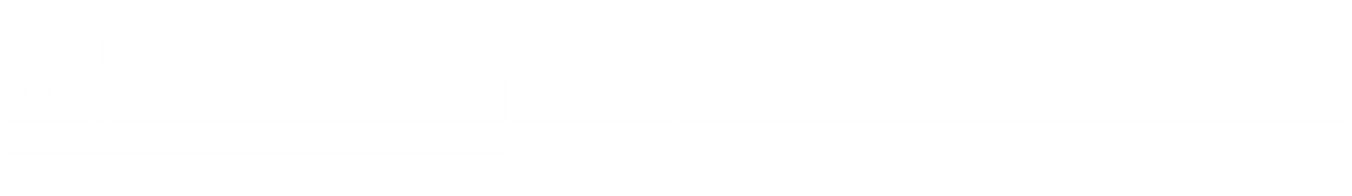
A **Mapped Address** is used when just the **receiver** does not support IPv6. This is exactly the same as a Compatible Address, except an extra **16 bits** before the actual IPv4 address are set to .



Again, we use **mixed representation**, so an example of a Mapped Address could be .

### Unique Local Unicast Address

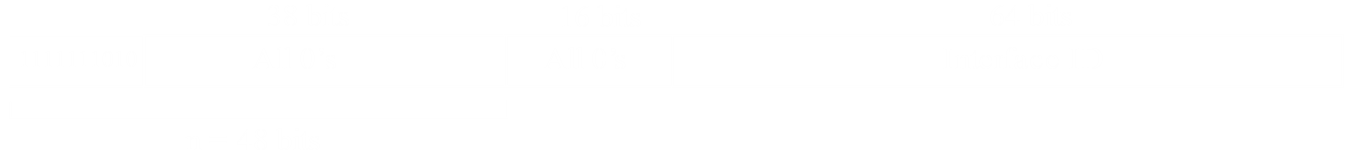
As mentioned above, a **Unique Local Unicast Address** is essentially a **private address**.



Here, the right-most 64 bits, the **Interface ID**, is essentially the physical address. MAC addresses are 48 bits, but there is a different format, called the Extended Unique Identifier (EUI) – 64, which uses 64 bits.

In the left-most 64 bits, we have a **7-bit prefix**, the 8th bit is either a or a as decided by the assigning authority, the next bits is a **random number** generated by the device and finally, we have a 16 bit **subnet ID**. Because of this, it means that a single organization can have subnets, which is more than 65000 subnets.

### Link Local Address



A **Link Local Address** is essentially an address used inside a private network, which is why all that is needed is the **prefix** to identify it and the **Interface ID** to identify devices.

The format has been chosen to make it compatible with **Unique Global Addresses**, which we will be looking into shortly.

### Multicast Address

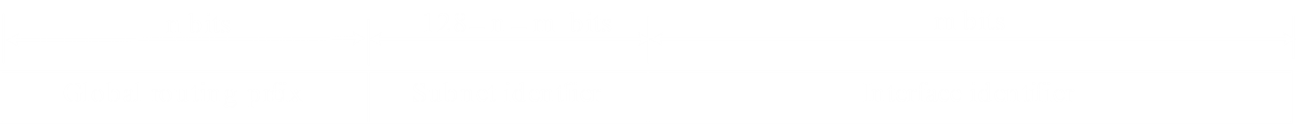


In a **Multicast Address**, we have an 8-bit prefix, all s, used to identify the address. Multicast addresses identify **groups** of addresses, as defined by the **Group ID**.

Groups may be **Permanent** or **Transient**, with the former being defined by the Internet authority and accessible at all times and the latter being temporary. The group type is identified using the 4-bit **Flag**.

We also have a 4-bit **Scope**, which identifies the scope of the group address, as shown in the figure.

## 26.3 Global Unicast Addresses



A **Global Unicast Address** has an -bit **Global Routing Prefix**, which can be used to identify the network on the global internet. The -bit **Interface Identifier** is used to identify specific hosts within the network. In between these two, we have the **Subnet Identifier**, which has the size bits.

Generally, and , which makes the subnet sizes bits. Notice that this makes **Link Local Addresses** perfectly compatible with this format.

If an organization is assigned the block , its subnets would be , and so on.

### Interface Identifier

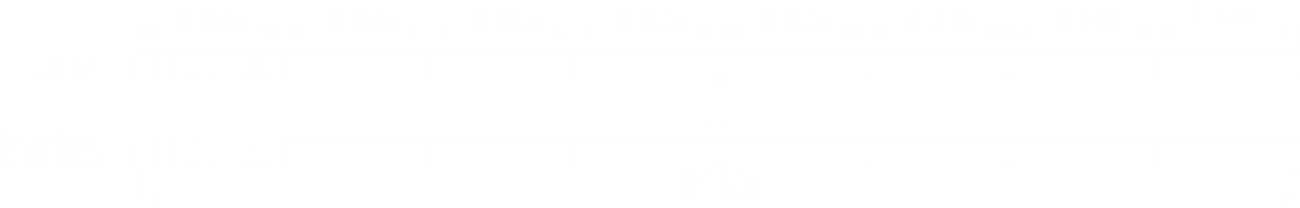
As mentioned before, the **Interface Identifier** is just the physical address of the machine. This brings up the question of why we don’t just use the physical address itself, since it is unique for literally every device in the world.

With IPv4, it was possible to argue that IPv4 addresses were shorter than physical addresses and therefore less powerful routers were able to process them. However, this is no longer true for IPv6. Now, the argument becomes that IPv6 addresses (and IPv4 addresses as well) are **hierarchical**, whereas physical addresses are **flat**.

Essentially, there are levels of groupings in IP addresses that create a situation where no router in the world needs to store all possible IP addresses. They simply store the network addresses and some default address to which they send the packets if the network address required is not in their own routing table. If we used physical addresses, every router in the world would have to store potentially every physical address in the world, making it virtually impossible to find a single one.

### Mapping EUI-64 Addresses

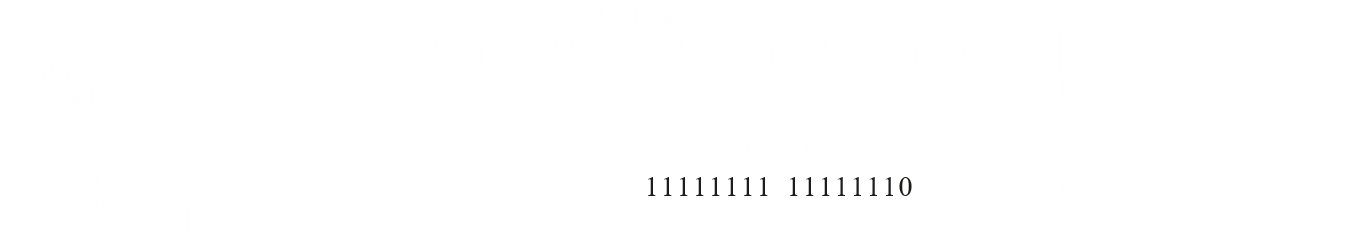
To map an **EUI-64 address** to an Interface Identifier address, we simply need to change the **Global/Local Bit** from to . This is the second bit in the left-most octet of the EUI-64 address.



For an EUI-64 address , we convert () to (), thus producing .

### Mapping Ethernet MAC Addresses

To map an **Ethernet MAC address** to an Interface Identifier address, we first change the **Global/Local Bit** from to in the left-most octet. Then, in between the third and fourth octets, we **add 16 bits**, , or 15 s followed by a .



For the Ethernet MAC address , we first change the Global/Local Bit, so becomes , and then we add in between and , thus making the complete Interface Identifier .

In conclusion, if we have an organization that has been assigned the block and we have a device with the Ethernet MAC in the third subnet, the complete IPv6 Address would be:

## 26.4 Autoconfiguration

In IPv4, hosts and routers are **configured manually**. However, using the **Dynamic Host Configuration Protocol** (DHCP), it is possible to allocate an IPv4 address to any new host that joins a network.

In IPv6, DHCP can still be used, but a host can also configure itself. This is called **Autoconfiguration**.

1. The host creates a **link local address** by taking the 10-bit link local prefix (), adding 54 s behind it and then adding its own 64-bit interfaces identifier, generated from its own physical address.
2. The link local address generated should be unique, since the physical address of the device is supposed to be unique. Just in case it is not though, the new host sends out a **neighbour solicitation message** and waits for a **neighbour advertisement message**. It there is another host with the same link local address, the process fails. The new host must be configured using DHCP.
3. If the link local address is unique, the host next needs a **global unicast address**. For this, it sends out a **router solicitation message** to which the router responds with a **router advertisement message**, which includes the **global unicast prefix** and **subnet prefix** which the host needs to add to its interface identifier to generate the global unicast address. If the router cannot help for some reason, it informs the host using a flag in the router advertisement message, at which point other mechanisms for configuration are used.