Engineer

IPv6

the version of the Internet Protocol with 128-bit addresses

What it is

IPv6, written out as Internet Protocol version 6, is the version of the Internet Protocol designed as the successor to IPv4, the version described in RFC 791. Its current specification is RFC 8200 from July 2017, an Internet Standard (STD 86) that replaced RFC 2460. The change everyone notices is the size of an address: 128 bits instead of IPv4's 32. The whole of IPv4 has 4,294,967,296 addresses; a single IPv6 subnet of the usual size, a /64, holds that number squared.

The rest of the packet changed too. RFC 8200 drops some IPv4 header fields or makes them optional to cut processing cost, carries optional information in separate extension headers placed after the main one, and renames Time to Live to Hop Limit, which every forwarding node lowers by one. There is no header checksum. Routers never split a packet: fragmentation in IPv6 is done only by the source, every link must carry packets of at least 1280 octets, and senders are strongly encouraged to use Path MTU Discovery to find out how large a packet a path will carry.

An address is written as eight groups of 16 bits in hexadecimal, separated by colons. RFC 4291 lets you drop the leading zeros in a group and replace one run of all-zero groups with a double colon, which may appear only once, so 2001:db8:0:0:8:800:200c:417a can be written 2001:db8::8:800:200c:417a. Because that leaves many ways to write the same address, RFC 5952 from August 2010 picks one: lower case, no leading zeros, and the double colon on the longest run of zero groups (the first one if two are equally long), never on a single group. Apart from special addresses such as loopback and link-local ones, every example address and prefix in this article comes from a documentation range: 2001:db8::/32, which RFC 3849 sets aside, or for IPv4 the blocks reserved by RFC 5737.

RFC 4291 also changes what an address is attached to. Addresses belong to interfaces, not to machines, and one interface can have several at once. Every interface must have a link-local address under fe80::/10, valid only on its own link. The loopback address is ::1. There is no broadcast: multicast does that job, and ff02::1 reaches all nodes on the link. Unique local addresses under fc00::/7, defined in RFC 4193, serve inside a site. Global unicast addresses are used for communication beyond the local site. For most addresses the last 64 bits are the interface identifier, which is why an ordinary network segment is a /64.

Hosts usually number themselves. Under stateless address autoconfiguration, RFC 4862, routers announce the prefixes of a link in periodic Router Advertisements, each host combines a prefix with its own interface identifier, and before using the result it runs Duplicate Address Detection, sending Neighbor Solicitation messages to check that no other node holds the address. DHCPv6 is the alternative where a site wants tighter control over exact assignments, and the two can run side by side.

Use the concept in a decision

Put “What is IPv6?” to work in a real decision. We can help relate the concept to the system you are building or buying. Wi-Fi, firewalls and remote access set up so staff can work from anywhere and only the right people get in.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Managed 30 Ubuntu Linux VPS instances — 30 Linux servers managed, with recovery planned for

Managed 30 Ubuntu Linux VPS instances Get in touch →

Why it matters

For many teams IPv6 is not a project they decide to start. The point of RFC 4862 is that a host needs nobody to type an address into it: put a server on a link where a router advertises a prefix, and it can pick up a global IPv6 address by itself. Any allow-list, firewall rule, log parser or health check written with only IPv4 in mind then has a blind spot of exactly the size of the new address space.

That blind spot is easy to miss, because the two versions are separate protocols that share nothing on the wire. A packet filter matches their headers separately, a rate limit keyed on IPv4 addresses never sees an IPv6 client, and a test that only connects to 127.0.0.1 says nothing about ::1. A host that is locked down on one version and open on the other has, in practice, no firewall.

IPv6 also leans on its control messages more than IPv4 does. Neighbour discovery, router advertisements and Path MTU Discovery all travel as ICMPv6, so a habit carried over from IPv4, dropping ICMP at the edge, breaks IPv6 in ways that may only show up once a connection starts moving data. And because one address has many valid spellings, code that compares addresses as strings will disagree with itself.

What it looks like in practice

Python's standard `ipaddress` module parses every valid spelling and prints a short form, with leading zeros dropped and the longest run of zero groups collapsed, so it is a quick way to plan a prefix and to check that two spellings are the same address. The session below is this article's illustration, run on Python 3.15 with addresses from the documentation prefix:

```
>>> import ipaddress
>>> site = ipaddress.ip_network("2001:db8:42::/48")
>>> site.num_addresses == 2**80
True
>>> lans = site.subnets(new_prefix=64)
>>> next(lans), next(lans)
(IPv6Network('2001:db8:42::/64'), IPv6Network('2001:db8:42:1::/64'))
>>> len(list(site.subnets(new_prefix=64)))
65536
>>> a = ipaddress.ip_address("2001:0DB8:0000:0000:0001:0000:0000:0001")
>>> a
IPv6Address('2001:db8::1:0:0:1')
>>> a.exploded
'2001:0db8:0000:0000:0001:0000:0000:0001'
>>> a == ipaddress.ip_address("2001:db8:0:0:1::1")
True
>>> ipaddress.ip_address("fe80::1%eth0").scope_id
'eth0'
```

A /48 leaves 80 bits free, and split into /64 segments it gives 65,536 of them, numbered by the fourth group. The address typed in upper case with every zero spelled out comes back in the short form, 2001:db8::1:0:0:1, where the double colon covers the first of two equally long runs of zeros, and it compares equal to 2001:db8:0:0:1::1, a different but valid spelling of the same 128 bits. The last line shows a link-local address carrying its zone, the `%eth0` that says which link it belongs to.

On a Linux host running nftables, the second part of this illustration is a rule set that admits SSH from an office over both versions. The nft(8) manual describes `inet` as a hybrid family whose tables see IPv4 and IPv6 packets alike, while `ip saddr` reads the IPv4 header and `ip6 saddr` the IPv6 one:

```
add table inet filter
add chain inet filter input { type filter hook input priority filter; }
add rule inet filter input ip saddr 198.51.100.0/24 tcp dport 22 accept
add rule inet filter input ip6 saddr 2001:db8:42::/48 tcp dport 22 accept
add rule inet filter input tcp dport 22 drop
```

Without the fourth line, an administrator connecting from the office's IPv6 prefix matches neither accept rule and is dropped, which at least fails loudly. The quieter failure is the reverse: a rule set built in the `ip` family alone, on a host that also has an IPv6 address, never looks at IPv6 packets at all.

The portfolio entry Provisioned a second server from code, from September 2026, deals with the same exposure from the provisioning side. The first machine had been created by hand and given its firewall afterwards, which left a window in which a fresh host with a default password policy was reachable from the internet. The second is created only after its firewall: the firewall play runs before the server play, and both providers' firewalls are written directly against their REST APIs. That page does not mention IPv6, so the rules above are this article's illustration and not that host's. What this article adds is that the firewall has to cover every address the host will have, IPv6 included, before the host exists.

Where it goes wrong

A firewall that only knows IPv4. Rules written in the `ip` family, or allow-lists holding only IPv4 prefixes, leave every IPv6 address of the host to whatever comes next. Write the rule set in the `inet` family or keep `ip` and `ip6` tables in step, pair every `ip saddr` rule with an `ip6 saddr` rule, and test each service by connecting to its IPv6 address as well as its IPv4 one.

Dropping ICMPv6 wholesale. RFC 4890, on filtering these messages in firewalls, puts it plainly: "ICMPv6 is essential to the functioning of IPv6". A router that cannot forward a packet because it is too large for the next link discards it and returns a Packet Too Big message, and RFC 8201 notes that if filtering stops those messages, the source never learns the path MTU. The result is what it calls

a black-hole connection: the TCP handshake succeeds, then the transfer hangs. RFC 4890 also lists Neighbor Solicitation and Advertisement and Router Solicitation and Advertisement as messages a node must not drop if it is to take part in an IPv6 network. Accept the types nft calls `packet-too-big`, `nd-neighbor-solicit`, `nd-neighbor-advert`, `nd-router-solicit` and `nd-router-advert` before any broader drop.

Comparing addresses as text. RFC 5952 gives the example of a spreadsheet search for 2001:db8:0:0:1::1 that misses an entry written 2001:db8::1:0:0:1, which matters when the list is searched before a new address is assigned, to check that it is not already in use. Parse addresses before comparing them, and store and print one canonical form, such as the short form Python's `ipaddress` returns from `str()`.

An address and a port run together. 2001:db8::1:80 could be an address, or an address followed by port 80; RFC 5952 calls that form ambiguous and recommends square brackets, [2001:db8::1]:80, as in URLs. Always write the brackets, and in code split a host and port on the closing bracket rather than on the last colon.

A link prefix that is not a /64. Autoconfiguration needs the prefix and the interface identifier to add up to 128 bits, and with 64-bit identifiers RFC 4862 has a host ignore an advertised prefix of any other length, so hosts on a link whose Router Advertisements carry a /72 or a /56 form no address from that prefix. Plan each link as a /64 and split larger allocations into /64s, as the session above does.

Link-local addresses without their zone. Every interface has an address in fe80::/10, and RFC 4291 limits those addresses to a single link, so on a host with several interfaces the address alone does not say which link is meant. Python's documentation notes that two addresses differing only in their zone are not equal. Keep the zone with the address (fe80::1%eth0), or give services a global or unique local address and keep link-local ones for the link.

Further reading

- RFC 8200: Internet Protocol, Version 6 (IPv6) Specification — the current IPv6 standard, with the header format, extension headers and the rules for fragmentation and packet size.

- RFC 4291: IP Version 6 Addressing Architecture — address types, the text forms, link-local and multicast addresses, and 64-bit interface identifiers.
- RFC 5952: A Recommendation for IPv6 Address Text Representation — the canonical way to write an address, and the trouble that many spellings cause.
- RFC 4862: IPv6 Stateless Address Autoconfiguration — how hosts form their own addresses from advertised prefixes and check them for duplicates.
- RFC 4890: Recommendations for Filtering ICMPv6 Messages in Firewalls — which ICMPv6 messages a firewall must let through, and why.
- ipaddress: IPv4/IPv6 manipulation library — Python's reference for parsing, comparing and splitting IPv6 addresses and networks.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is IPv6?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/ipv6/