Engineer

IP address

a number that tells the network where to deliver a packet

What it is

An IP address is the number the Internet Protocol writes into every packet to say where the packet is going, next to a second address that says where it came from. RFC 791, the 1981 specification of IP version 4, calls it an internet address and draws the line in one sentence: “An address indicates where it is.” A name, such as a domain, says what you are looking for and has to be turned into an address first, which is the job of DNS. The route, the path between the two addresses, is left to the routers in between.

In IPv4 the address is 32 bits, four octets, written as four decimal numbers from 0 to 255 separated by dots: 192.0.2.10. RFC 791 split those bits into a network number and a local part, with three fixed classes deciding where the split fell. The classes fitted most organisations badly: a class C network held at most 254 hosts, a class B up to 65,534, so class B numbers ran short while routing tables grew. In September 1993 RFC 1519 replaced the classes with CIDR, which lets the split fall after any bit and writes its position after a slash. 192.0.2.0/24 is the block whose first 24 bits are fixed: 256 addresses.

IPv6 widens the address to 128 bits. It is written as eight groups of up to four hexadecimal digits separated by colons, and RFC 4291 lets one run of zero groups collapse to `::`, so 2001:db8:0:0:0:0:2:1 can also be written 2001:db8::2:1. Because the same address could then be spelt many ways, RFC 5952 later fixed one recommended form: lower case, no leading zeros in a group, and `::` used for the longest run of zero groups.

An address belongs to a network interface, not to a machine. RFC 791 already said that a single host must be able to act as several, with several internet addresses, and that some hosts have several physical interfaces. RFC 4291 requires every IPv6 interface to have at least one link-local address, valid only on

its own network segment, whatever other addresses it holds. A host on a network that runs both versions therefore carries several addresses at once, besides the loopback addresses it uses to reach itself: the 127.0.0.0/8 block in IPv4 and `::1` in IPv6.

Not every address is reachable from everywhere. RFC 1918 sets aside three IPv4 blocks for private networks, 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16, and says that routing information about them shall not be passed between organisations. A device holding one therefore reaches the internet only through something that stands in for it, such as NAT, which RFC 3022 describes as translating the private address into a public one on the way out. Three further blocks, 192.0.2.0/24, 198.51.100.0/24 and 203.0.113.0/24, are reserved by RFC 5737 for documentation, and RFC 3849 does the same for 2001:db8::/32 in IPv6. Every example address in this article comes from those documentation ranges.

Use the concept in a decision

Put “What is an IP address?” 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

The address is the one fact about a client that a server always has, because without it the reply would have nowhere to go. That makes it the default key for many decisions: firewall rules allow or block by address, a rate limit counts requests per address, an access log writes it on every line, and an abuse report names it. Each of these works only as well as the address identifies what you think it identifies.

It identifies less than it seems to. Behind NAT, every device in an office or a household can reach the internet from the same public IPv4 address, so a block or a limit aimed at one of them lands on all of them. Behind a proxy or a CDN, the server sees the proxy’s address on every request, and the client’s own address arrives, if at all, in a header that someone else wrote. A rule written against an address is a rule about a network position, never about a person.

At the same time it reveals more than people expect. RFC 7239, which standardised the `Forwarded` header that proxies use to pass the client's address on, notes in its privacy section that an IP address may identify a client uniquely, show which operator it uses, and give a rough idea of where it is. A log full of addresses is therefore a liability as well as a tool, and a good reason to decide what you keep before you start keeping it.

And there are now two kinds. Anything that parses, stores or compares addresses has to accept IPv4 and IPv6 alike, because a client on a network that offers both may arrive over either. Code written when every address fit in fifteen characters of digits and dots breaks on the first IPv6 visitor, usually somewhere far from where the address was read.

What it looks like in practice

Python's standard library includes an `ipaddress` module that parses both versions and does the arithmetic, so you never have to split strings by hand. In an interactive session:

```
>>> import ipaddress
>>> ipaddress.ip_address("203.0.113.57")
IPv4Address('203.0.113.57')
>>> ipaddress.ip_address("2001:0DB8:0000:0000:0000:0000:0002:0001")
IPv6Address('2001:db8::2:1')
>>> ipaddress.ip_network("203.0.113.57/24", strict=False)
IPv4Network('203.0.113.0/24')
>>> ipaddress.ip_network("2001:db8:42:7::5/48", strict=False)
IPv6Network('2001:db8:42::/48')
```

The second call shows the normalisation that RFC 5952 asks for: the long upper-case spelling and the short lower-case one are the same address, and the module prints the canonical form. A string such as 192.0.2.300 is refused with a `ValueError`, because 300 does not fit in an octet. The last two calls turn an address into the network around it. Without `strict=False` the module would refuse those as well, because 203.0.113.57/24 has host bits set; with it, the host part is set to zero and what remains is the network alone.

That zeroing is how this site's web access log treats visitors' addresses. Its privacy page says the address is shortened as the web log line is written, with the last part replaced by zeroes before the line reaches disk, so the stored value

identifies a rough network in which a whole office or street can look identical. The server records the address under two names, masks both, and drops headers that can carry an address of their own before writing the line. The page does not say how many bits survive; the /24 and /48 above are this article's illustration, not the site's settings. Masking at write time keeps full addresses out of that web log without relying on a later clean-up job. The same privacy page separately records a Gopher exception: the operating system's per-connection log names the source address and expires after thirty days.

Where it goes wrong

Trusting a header for the client's address. `X-Forwarded-For` and the standard `Forwarded` header are plain request headers, and RFC 7239 warns that any hop between the client and the server, and the client itself, may rewrite the `Forwarded` header, by mistake or on purpose. Code that takes the first address in the list lets any client choose its own address, and with it its own rate limit and its own place on an allow-list. Use the address of the TCP connection unless the request came through a proxy you run, and then read only what your own proxies appended, counting from the right.

Treating an address as a person. Because many users can share one address, a ban on an address may lock out an entire office, and a limit sized for one person throttles a whole school. Use addresses for coarse decisions, set limits with shared addresses in mind, and tie anything personal to an authenticated account instead.

Comparing addresses as strings. 2001:db8::2:1 and 2001:DB8:0:0:0:0:2:1 are one address and two different strings, so an allow-list matched by text accepts one and rejects the other. Parse both sides before comparing, or store only the RFC 5952 form.

Parsing for IPv4 only. A regular expression for four dotted numbers, a column sized for fifteen characters, or code that splits host and port at the last colon will all fail on IPv6. Use a real parser and a column type or length meant for both versions. When a port is written next to an IPv6 address, RFC 5952 recommends brackets, as in [2001:db8::1]:443, because without them the last group could be read as part of the address.

Keeping full addresses longer than needed. Logs, analytics tables and backups collect addresses for years if nothing stops them. Decide what precision you need, shorten the address when it is written rather than afterwards, and give the remainder an expiry date.

Using real addresses in examples. A sample configuration, a test fixture or a screenshot that uses someone's real address sends copied traffic to a stranger. RFC 5737 reserves its three blocks precisely to reduce that kind of conflict and confusion; use them, and 2001:db8::/32 for IPv6.

Further reading

- RFC 791: Internet Protocol — the 1981 specification of IPv4, with its 32-bit address and the distinction between names, addresses and routes.
- RFC 4291: IP Version 6 Addressing Architecture — the IPv6 address types, their text forms and the link-local requirement.
- RFC 5952: A Recommendation for IPv6 Address Text Representation — one canonical way to write an IPv6 address, and the trouble that flexible spellings cause.
- RFC 7239: Forwarded HTTP Extension — how proxies pass a client's address on, why the result cannot simply be trusted, and its privacy cost.
- RFC 5737: IPv4 Address Blocks Reserved for Documentation — the three ranges meant for examples.
- ipaddress: IPv4/IPv6 manipulation library — Python's standard module for parsing, comparing and grouping addresses.



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 an IP address?” 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/ip-address/