IPv6 address

```
^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`(`	start of a capturing group
`(`	start of a capturing group
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`:`	literal text ":"
`){7}`	end of group, exactly 7 times
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`|`	OR — try the alternative
`(`	start of a capturing group
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`:`	literal text ":"
`){1,7}`	end of group, between 1 and 7 times
`:`	literal text ":"
`|`	OR — try the alternative
`(`	start of a capturing group
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`:`	literal text ":"
`){1,6}`	end of group, between 1 and 6 times
`:`	literal text ":"
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`|`	OR — try the alternative
`(`	start of a capturing group
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`:`	literal text ":"
`){1,5}`	end of group, between 1 and 5 times

© 2026 regexguide.com · a human-curated regex explainer & pattern library
Disclaimer: provided as-is, without warranty — test against your own data before production use.

Token	Meaning
(	start of a capturing group
:	literal text ":"
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
){1,2}	end of group, between 1 and 2 times
|	OR — try the alternative
(	start of a capturing group
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
:	literal text ":"
){1,4}	end of group, between 1 and 4 times
(	start of a capturing group
:	literal text ":"
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
){1,3}	end of group, between 1 and 3 times
|	OR — try the alternative
(	start of a capturing group
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
:	literal text ":"
){1,3}	end of group, between 1 and 3 times
(	start of a capturing group
:	literal text ":"
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
){1,4}	end of group, between 1 and 4 times
|	OR — try the alternative
(	start of a capturing group
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
:	literal text ":"
){1,2}	end of group, between 1 and 2 times
(	start of a capturing group
:	literal text ":"
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
){1,5}	end of group, between 1 and 5 times
|	OR — try the alternative
[0-9a-fA-F]{1,4}	between 1 and 4 times: any of: digits, a–f, A–F
:	literal text ":"
(	start of a capturing group
:	literal text ":"

© 2026 regexguide.com · a human-curated regex explainer & pattern library

Disclaimer: provided as-is, without warranty — test against your own data before production use.

Token	Meaning
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`){1,6}`	end of group, between 1 and 6 times
`|`	OR — try the alternative
`:`	literal text ":"
`(`	start of a capturing group
`(`	start of a capturing group
`:`	literal text ":"
`[0-9a-fA-F]{1,4}`	between 1 and 4 times: any of: digits, a–f, A–F
`){1,7}`	end of group, between 1 and 7 times
`|`	OR — try the alternative
`:`	literal text ":"
`)`	end of group
`)`	end of group
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$/;
re.test(value);
```

Python

```
import re
re.match(r"^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$", value)
```

Java

```
Pattern.compile("^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$");
```

Go

```
regexp.MustCompile(`^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$`).MatchString(value)
```

Ruby

```
/^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$/.match?(value)
```


© 2026 regexguide.com · a human-curated regex explainer & pattern library

Disclaimer: provided as-is, without warranty — test against your own data before production use.

PHP

```
preg_match('~^(([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(:[0-9a-fA-F]{1,4}){1,6}|:((:[0-9a-fA-F]{1,4}){1,7}|:))$~', $value);
```

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Case sensitivity.** Letter ranges are case-sensitive — use the i flag if the input case can vary.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the ipv6 address against a source of truth (database, API, or checksum) where it matters.

© 2026 regexguide.com · a human-curated regex explainer & pattern library

Disclaimer: provided as-is, without warranty — test against your own data before production use.