French phone (+33)

French phone with +33 prefix.

```
^\+33[\s\-]?[1-9](\s?\d{2}){4}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`\+`	literal “+”
`33`	literal text “33”
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`[1-9]`	any of: 1–9
`(`	start of a capturing group
`\s?`	optional (zero or one): whitespace
`\d{2}`	exactly 2 times: digit (0–9)
`){4}`	end of group, exactly 4 times
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^\+33[\s\-]?[1-9](\s?\d{2}){4}$/;
re.test(value);
```

Python

```
import re
re.match(r"^\+33[\s\-]?[1-9](\s?\d{2}){4}$", value)
```

Java

```
Pattern.compile("^\\+33[\\s\\-]?[1-9](\\s?\\d{2}){4}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^\+33[\s\-]?[1-9](\s?\d{2}){4}$");
```

Go

```
regexp.MustCompile(`^\+33[\s\-]?[1-9](\s?\d{2}){4}$`).MatchString(value)
```

Ruby

```
/^\+33[\s\-]?[1-9](\s?\d{2}){4}$/.match?(value)
```

PHP

```
preg_match('~^\+33[\s\-]?[1-9](\s?\d{2}){4}$~', $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.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the french phone (+33) against a source of truth (database, API, or checksum) where it matters.