Premium-rate 1900 number

1900-series premium-rate services (paid by caller at higher rate).

```
^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`1900`	literal text “1900”
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`\d{2,3}`	between 2 and 3 times: digit (0–9)
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`\d{4,7}`	between 4 and 7 times: digit (0–9)
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$/;
re.test(value);
```

Python

```
import re
re.match(r"^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$", value)
```

Java

```
Pattern.compile("^1900[\\s\\-]?\\d{2,3}[\\s\\-]?\\d{4,7}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$");
```

Go

```
regexp.MustCompile(`^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$`).MatchString(value)
```

Ruby

```
/^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$/.match?(value)
```

PHP

```
preg_match('~^1900[\s\-]?\d{2,3}[\s\-]?\d{4,7}$~', $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.
- **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 premium-rate 1900 number against a source of truth (database, API, or checksum) where it matters.