Australian phone (+61)

Australian phone with +61 prefix.

```
^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`(`	start of a capturing group
`\+`	literal “+”
`61`	literal text “61”
`|`	OR — try the alternative
`0`	literal text “0”
`)`	end of group
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`[2-478]`	any of: 2–4, “7”, “8”
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`\d{4}`	exactly 4 times: digit (0–9)
`[\s\-]?`	optional (zero or one): any of: whitespace, “-”
`\d{4}`	exactly 4 times: digit (0–9)
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$/;
re.test(value);
```

Python

```
import re
re.match(r"^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$", value)
```

Java

```
Pattern.compile("^(\\+61|0)[\\s\\-]?[2-478][\\s\\-]?\\d{4}[\\s\\-]?\\d{4}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$");
```

Go

```
regexp.MustCompile(`^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$`).MatchString(value)
```

Ruby

```
/^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{4}$/.match?(value)
```

PHP

```
preg_match('~^(\+61|0)[\s\-]?[2-478][\s\-]?\d{4}[\s\-]?\d{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 australian phone (+61) against a source of truth (database, API, or checksum) where it matters.