

UPI ID (VPA)

Unified Payments Interface ID — username@psp.

```
^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}$
```

Token-by-token breakdown

Token	Meaning
^	start of string (or line in multiline mode)
[a-zA-Z0-9.\-_]{2,256}	between 2 and 256 times: any of: lowercase letters, uppercase letters, digits, ".", "-", "_"
@	literal text "@"
[a-zA-Z]	any of: lowercase letters, uppercase letters
[a-zA-Z]{2,64}	between 2 and 64 times: any of: lowercase letters, uppercase letters
\$	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	<pre>const re = /^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$/; re.test(value);</pre>
Python	<pre>import re re.match(r"^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$", value)</pre>
Java	<pre>Pattern.compile("^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$").matcher(value).matc hes();</pre>
C# / .NET	<pre>Regex.IsMatch(value, @"^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$");</pre>
Go	<pre>regexp.MustCompile(`^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$`).MatchString(value)</pre>
Ruby	<pre>/^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$/ .match?(value)</pre>
PHP	<pre>preg_match('~^[a-zA-Z0-9.\-_]{2,256}@[a-zA-Z][a-zA-Z]{2,64}\$~', \$value);</pre>

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.
- **ASCII letters only.** [a-zA-Z] excludes accented and non-Latin letters (é, ü, ß, ñ, and non-Latin scripts). For international input use Unicode properties like \p{L} with the u 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 upi id (vpa) against a source of truth (database, API, or checksum) where it matters.