

Split

Der `-split` Operator in PowerShell wird verwendet, um einen String anhand eines Trennzeichens (Delimiter) in ein Array von Teilstrings aufzuteilen.

□ Syntax

```
<string> -split <delimiter>
```

Oder mit erweiterten Optionen:

```
<string> -split <delimiter>, <max-substrings>, <options>
```

□ Grundlagen

- Gibt immer ein **Array** (`System.String[]`) zurück
 - Der **Delimiter** ist ein **regulärer Ausdruck (Regex!)**, kein einfacher Text
 - Groß-/Kleinschreibung kann beeinflusst werden
-

□ Beispiele

Einfaches Splitten

```
$text = "Apfel,Birne,Banane"  
$result = $text -split ","
```

Ergebnis:

```
Apfel  
Birne  
Banane
```

Split mit Leerzeichen

```
$text = "Das ist ein Test"  
$result = $text -split " "
```

Mehrere Trennzeichen (Regex)

```
$text = "Apfel;Birne,Banane"  
$result = $text -split "[,;]"
```

Erklärung:

- `[ , ; ]` bedeutet: splitte bei Komma **oder** Semikolon

☐☐ Verhalten bei `max-substrings`

Wenn die maximale Anzahl an Elementen erreicht ist:

“ **Das letzte Element im Ergebnis enthält den gesamten verbleibenden Rest des Strings und wird nicht weiter gesplittet.**

```
$text = "A,B,C,D"  
$result = $text -split ",", 2
```

Ergebnis:

```
A  
B,C,D
```

Erklärung:

- Es werden maximal **2 Elemente** erzeugt
- Das erste Element entsteht durch den ersten Split
- Das zweite Element enthält **alles, was danach noch übrig ist** (B,C,D)

Split mit Optionen

```
$text = "a,b,c"  
$result = $text -split ",", 0, "IgnoreCase"
```

Mögliche Optionen:

- IgnoreCase
→ Groß- und Kleinschreibung wird ignoriert
→ "A,B,C" -split "a" würde trotzdem funktionieren
- CaseSensitive
→ Groß- und Kleinschreibung wird beachtet
→ "A,B,C" -split "a" liefert **kein Ergebnis**, da kein Match
- SimpleMatch
→ Der Delimiter wird **als normaler Text behandelt, nicht als Regex**
→ Sonderzeichen wie ., *, + verlieren ihre Regex-Bedeutung

Beispiel:

```
"1.2.3" -split ".", 0, "SimpleMatch"
```

→ funktioniert ohne Escape (\.)

⚠ Wichtige Hinweise

1. Regex-Falle

```
$text = "1.2.3"  
$result = $text -split "."
```

Problem:

. bedeutet im Regex „beliebiges Zeichen“

→ Ergebnis: komplett zerlegt

Lösung:

```
$result = $text -split "\"
```

2. Leere Elemente

```
"text,,text" -split ",,"
```

Ergebnis:

```
text
```

```
text
```

Erklärung:

- Jeder Teilstring entsteht **zwischen zwei Trennzeichen (Delimiter)**
- Wenn zwei Delimiter direkt aufeinander folgen (,,), liegt dazwischen **kein Inhalt**
- PowerShell erzeugt dafür trotzdem ein Element im Array

⚡ Dieses Element ist ein **leerer String** (""), kein `null`

Alternative Methoden

.Split() (kein Regex!)

```
$text = "A,B,C"  
$result = $text.Split(",")
```

Unterschied zu `-split`:

- `.Split()` verwendet **kein Regex**
- oft schneller und einfacher
- weniger flexibel

```
[regex]::Split()
```

```
[regex]::Split("A,B,C", ",")
```

→ Alternative mit explizitem Regex-Handling

Typische Anwendungsfälle

- CSV-Daten zerlegen
- Log-Dateien parsen
- Benutzereingaben aufteilen
- Pfade oder Listen verarbeiten

Mini-Beispiel aus der Praxis

```
$csv = "Max;Mustermann;30"
$name = $csv -split ";"

$vorname = $name[0]
$nachname = $name[1]
$alter = $name[2]
```

Merksatz

“-split“ denkt in Regex. Wenn du das vergisst, bekommst du Chaos statt Struktur.

Revision #1

Created 2026-04-17 22:27:15 UTC by John-Andreas Borinas

Updated 2026-08-13 11:28:08 UTC by John-Andreas Borinas