Start-Sleep

Das Cmdlet `Start-Sleep` pausiert die Ausführung eines Skripts oder einer Befehlssequenz für eine definierte Zeitspanne.

Es wird häufig verwendet, um:

- Wartezeiten einzubauen
- Prozesse zu synchronisieren
- Ressourcen zu schonen (z. B. bei Schleifen)

□□ Syntax

```
Start-Sleep [-Seconds] <Double> [<CommonParameters>]

Start-Sleep -Milliseconds <Int32> [<CommonParameters>]
```

□□ Parameter

`-Seconds`

- **Typ:** `Double`
- **Pflicht:** Ja (wenn `-Milliseconds` nicht verwendet wird)
- Gibt die Wartezeit in Sekunden an
- Dezimalwerte sind erlaubt

```
Start-Sleep -Seconds 2.5
```

`-Milliseconds`

- **Typ:** `Int32`
- **Pflicht:** Ja (wenn `-Seconds` nicht verwendet wird)

- Gibt die Wartezeit in Millisekunden an

```
Start-Sleep -Milliseconds 500
```

⚠ Hinweise zur Verwendung

- `-Seconds` und `-Milliseconds` **dürfen nicht gleichzeitig verwendet werden**
- Während der Ausführung blockiert `Start-Sleep` den aktuellen Thread vollständig
- Es erfolgt **keine Hintergrundverarbeitung** während der Pause

---

□□ Verhalten

Eigenschaft	Beschreibung
Blockierend	Ja
Rückgabewert	Keiner
Thread-Verhalten	Aktueller Thread wird pausiert
Genauigkeit	Abhängig vom System-Timer

□□ Beispiele

Einfaches Warten

```
Write-Host "Start"
Start-Sleep -Seconds 2
Write-Host "Ende"
```

Verwendung in Schleifen

```
for ($i = 1; $i -le 5; $i++) {
    Write-Host "Durchlauf $i"
    Start-Sleep -Seconds 1
}
```

Kurze Pause in Millisekunden

```
Start-Sleep -Milliseconds 200
```

Dynamische Wartezeit

```
$delay = 1.5
Start-Sleep -Seconds $delay
```

⚙ Typische Anwendungsfälle

- Polling (z. B. auf Datei oder Prozess warten)
- Debugging (Abläufe verlangsamen)
- UI-Skripte (z. B. kurz warten, bis Controls geladen sind)
- API-Rate-Limiting

□ Alternativen / Ergänzungen

`[System.Threading.Thread]::Sleep()`

```
[System.Threading.Thread]::Sleep(1000)
```

Unterschiede:

- Arbeitet nur mit Millisekunden
- Weniger PowerShell-idiomatisch
- Kein Support für `CommonParameters`

`Start-Sleep` vs. `Start-Job`

Szenario	Empfehlung
Einfaches Warten	`Start-Sleep`
Asynchrone Ausführung	`Start-Job`
UI-Responsiveness wichtig	Kein `Start-Sleep`

□□ Typische Fehler

1. Beide Parameter gleichzeitig verwenden

```
# □ Falsch
Start-Sleep -Seconds 1 -Milliseconds 500
```

---

2. UI einfrieren

```
# □ Problematisch in WinForms/WPF
Start-Sleep -Seconds 5
```

➡ Blockiert die UI komplett

---

3. Zu kurze Wartezeiten erwarten hohe Präzision

```
Start-Sleep -Milliseconds 1
```

➡ Systembedingt oft ungenau

□□ Best Practices

- Für UI-Anwendungen besser Timer verwenden statt `Start-Sleep`
- In Schleifen immer bewusst einsetzen, um CPU-Last zu reduzieren
- Wartezeiten so kurz wie möglich halten
- Für präzise Zeitsteuerung ggf. andere Mechanismen nutzen

Wenn du das Ding in einer WinForms-App benutzt, dann frierst du dir halt elegant dein UI ein und wunderst dich danach, warum alles tot wirkt. Klassischer Anfänger-Move, aber immerhin ein lehrreicher.

Revision #2
Created 2026-04-07 11:50:51 UTC by John-Andreas Borinas
Updated 2026-05-15 04:46:49 UTC by John-Andreas Borinas