4.3 Prozesse und Prozesszustände analysieren

Ein Betriebssystemdienst wird letztlich durch einen oder mehrere Prozesse ausgeführt. Auch wenn die Dienstverwaltung den Zustand „läuft“ meldet, kann der zugehörige Prozess blockiert sein, ungewöhnlich viele Ressourcen verbrauchen oder fortlaufend beendet und neu gestartet werden.

> **Grundsatz:**
> Ein vorhandener Prozess beweist nur, dass eine Prozessinstanz existiert. Er beweist nicht, dass der Dienst Anfragen korrekt verarbeitet.

Ziele dieser Seite

Nach dieser Seite sollst du:

- einen Dienst seinem Prozess zuordnen können,
- PID und Elternprozess bestimmen können,
- Benutzerkonto, Startzeit und Befehlszeile prüfen können,
- CPU- und Speicherverbrauch richtig bewerten können,
- Prozesszustände unter Windows, Linux und macOS einordnen können,
- Zombies und blockierte Prozesse erkennen können,
- Prozessneustarts und wechselnde PIDs feststellen können,
- Threads, Handles und geöffnete Dateien prüfen können,
- Diagnoseinformationen vor einer Prozessbeendigung sichern können.

---

1. Dienst, Prozess, Thread und PID unterscheiden

Begriff	Bedeutung
Dienst	Vom Betriebssystem verwaltete Hintergrundfunktion
Prozess	Laufende Instanz eines Programms
PID	Eindeutige Prozess-ID innerhalb des aktuell laufenden Systems
PPID	PID des Elternprozesses
Thread	Ausführungsstrang innerhalb eines Prozesses
Handle	Windows-Verweis auf eine verwendete Systemressource
Dateideskriptor	Unix-Verweis auf Datei, Socket, Pipe oder andere Ressource

Begriff	Bedeutung
Elternprozess	Prozess, der einen anderen Prozess erzeugt hat
Kindprozess	Von einem anderen Prozess erzeugter Prozess
Prozessbaum	Hierarchische Darstellung von Eltern- und Kindprozessen

Ein Dienst kann:

- genau einen Prozess verwenden,
- mehrere Prozesse starten,
- Kindprozesse erzeugen,
- mehrere Dienste in einem gemeinsamen Prozess ausführen,
- bedarfsgesteuert ohne dauerhaften Prozess arbeiten,
- nach einem Absturz automatisch eine neue PID erhalten.

> Eine PID ist nicht dauerhaft. Nach dem Ende eines Prozesses kann dieselbe Nummer später einem anderen Prozess zugewiesen werden.

---

2. Prozesse unter Windows suchen und anzeigen

Die Platzhalter `<PROZESS>` und `<PID>` müssen ersetzt werden.

Prozess nach Namen suchen:

```
[RO] Get-Process -Name "<PROZESS>" -ErrorAction SilentlyContinue
```

Die Dateiendung `.exe` wird bei `Get-Process -Name` normalerweise nicht angegeben.

Prozess über die PID suchen:

```
[RO] Get-Process -Id <PID> -ErrorAction SilentlyContinue
```

Ausgewählte Prozessinformationen anzeigen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
```

```
    ProcessName,
    StartTime,
    CPU,
    WorkingSet64,
    PrivateMemorySize64,
    VirtualMemorySize64,
    HandleCount,
    Responding
```

Alle Prozesse nach CPU-Zeit sortieren:

```
[RO] Get-Process |
  Sort-Object CPU -Descending |
  Select-Object -First 15 Id, ProcessName, CPU, WorkingSet64
```

Alle Prozesse nach belegtem Arbeitsspeicher sortieren:

```
[RO] Get-Process |
  Sort-Object WorkingSet64 -Descending |
  Select-Object -First 15 Id, ProcessName, WorkingSet64, CPU
```

Klassische Prozessübersicht:

```
[RO] tasklist
```

Bestimmten Prozess mit `tasklist` suchen:

```
[RO] tasklist /FI "PID eq <PID>"
```

> Eigenschaften wie `StartTime` oder `Path` können bei geschützten Prozessen oder ohne ausreichende Rechte einen Zugriffsfehler verursachen.

---

3. Dienst einem Windows-Prozess zuordnen

Dienstname, Status und PID ermitteln:

```
[RO] Get-CimInstance Win32_Service -Filter "Name='<DIENSTNAME>'" |
    Select-Object Name, DisplayName, State, StartName, ProcessId
```

Alle laufenden Dienste mit ihrer PID anzeigen:

```
[RO] Get-CimInstance Win32_Service |
    Where-Object State -eq "Running" |
    Select-Object Name, DisplayName, ProcessId, StartName |
    Sort-Object ProcessId
```

Alle Dienste innerhalb einer bestimmten PID suchen:

```
[RO] Get-CimInstance Win32_Service -Filter "ProcessId=<PID>" |
    Select-Object Name, DisplayName, State, StartName
```

Erweiterte Dienstabfrage mit `sc.exe`:

```
[RO] sc.exe queryex "<DIENSTNAME>"
```

Dienstinformationen mit Prozessinformationen verbinden:

```
[RO][SENS] $service = Get-CimInstance Win32_Service -Filter "Name='<DIENSTNAME>'"
Get-CimInstance Win32_Process -Filter "ProcessId=$($service.ProcessId)" |
    Select-Object ProcessId,
                  ParentProcessId,
                  Name,
                  ExecutablePath,
                  CommandLine
```

> Mehrere Windows-Dienste können sich einen gemeinsamen Hostprozess teilen. Das ist beispielsweise bei bestimmten `svchost.exe`-Instanzen üblich. Deshalb darf nicht allein anhand des Prozessnamens entschieden werden, welcher Dienst betroffen ist.

4. Windows-Prozessdetails mit CIM prüfen

Prozessname, Elternprozess, Pfad und Befehlszeile anzeigen:

```
[RO][SENS] Get-CimInstance Win32_Process -Filter "ProcessId=<PID>" |
    Select-Object ProcessId,
                  ParentProcessId,
                  Name,
                  ExecutablePath,
                  CommandLine,
                  CreationDate
```

Elternprozess ermitteln:

```
[RO] $process = Get-CimInstance Win32_Process -Filter "ProcessId=<PID>"
Get-CimInstance Win32_Process -Filter "ProcessId=$($process.ParentProcessId)" |
    Select-Object ProcessId, ParentProcessId, Name, CreationDate
```

Direkte Kindprozesse anzeigen:

```
[RO][SENS] Get-CimInstance Win32_Process -Filter "ParentProcessId=<PID>" |
    Select-Object ProcessId, ParentProcessId, Name, CommandLine
```

Anzahl der Threads und Handles prüfen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                  ProcessName,
                  HandleCount,
                  @{Name="ThreadCount";Expression={$_.Threads.Count}}
```

Ausführungskonto des Prozesses abfragen:

```
[RO][PRIV] Invoke-CimMethod `
    -InputObject (Get-CimInstance Win32_Process -Filter "ProcessId=<PID>") `
    -MethodName GetOwner
```

> [SENS]: Befehlszeilen können interne Pfade, Benutzernamen, Serveradressen oder unsicher übergebene Zugangsdaten enthalten. Die Ausgabe darf nicht ungeprüft weitergegeben werden.

---

5. Prozesse unter Linux suchen und anzeigen

Prozess nach Namen suchen:

```
[RO] pgrep -a "<PROZESS>"
```

Exakte Übereinstimmung des Prozessnamens verwenden:

```
[RO] pgrep -x -a "<PROZESS>"
```

Vollständige Befehlszeile durchsuchen:

```
[RO][SENS] pgrep -f -a "<SUCHMUSTER>"
```

Bestimmte PID untersuchen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,lstart,etime,stat,%cpu,%mem,rss,vsz,cmd
```

Alle Prozesse ausführlich anzeigen:

```
[RO][SENS] ps -eo user,pid,ppid,lstart,etime,stat,%cpu,%mem,rss,vsz,cmd
```

Nach CPU-Auslastung sortieren:

```
[RO] ps -eo pid,ppid,user,stat,%cpu,%mem,etime,comm \
  --sort=-%cpu |
  head -n 16
```

Nach residentem Speicher sortieren:

```
[RO] ps -eo pid,ppid,user,stat,%cpu,%mem,rss,etime,comm \
  --sort=-rss |
  head -n 16
```

Interaktive Prozessübersicht:

```
[RO] top
```

Falls installiert:

```
[RO] htop
```

> `htop` gehört nicht auf jeder Linux-Distribution zur Standardinstallation.

---

6. Linux-Prozessdetails über das proc-Dateisystem prüfen

Linux stellt Prozessinformationen normalerweise unter `/proc/<PID>` bereit.

Ausführbare Datei bestimmen:

```
[RO][PRIV] sudo readlink -f "/proc/<PID>/exe"
```

Arbeitsverzeichnis anzeigen:

```
[RO][PRIV] sudo readlink -f "/proc/<PID>/cwd"
```

Prozessstatus anzeigen:

```
[RO][FILE][SENS] cat "/proc/<PID>/status"
```

Befehlszeile lesbar darstellen:

```
[RO][FILE][SENS] tr '\0' ' ' < "/proc/<PID>/cmdline"
```

Umgebungsvariablen anzeigen:

```
[RO][FILE][SENS][PRIV] sudo tr '\0' '\n' < "/proc/<PID>/environ"
```

Grenzwerte des Prozesses anzeigen:

```
[RO][FILE] cat "/proc/<PID>/limits"
```

Eingehängte Ressourcen des Prozesses anzeigen:

```
[RO][FILE][SENS] cat "/proc/<PID>/mountinfo"
```

Anzahl geöffneter Dateideskriptoren ermitteln:

```
[RO][PRIV] sudo find "/proc/<PID>/fd" -mindepth 1 -maxdepth 1 -printf '.' |
  wc -c
```

> `/proc/<PID>/environ` kann Kennwörter, Token oder andere geheime Werte enthalten. Diese Abfrage darf nur bei berechtigtem Diagnosebedarf erfolgen. Die Ausgabe gehört nicht ungefiltert in ein Ticket.

---

7. systemd-Dienst einem Linux-Prozess zuordnen

Hauptprozess eines Dienstes anzeigen:

```
[RO] systemctl show "<DIENST>" \
  --property=MainPID,ControlPID,ExecMainPID,ExecMainStartTimestamp
```

Prozessbaum der Unit im Status anzeigen:

```
[RO] systemctl status "<DIENST>" --no-pager
```

Prozesse innerhalb der Unit beziehungsweise Control Group anzeigen:

```
[RO] systemd-cgls --unit "<DIENST>"
```

Ressourcennutzung von Control Groups beobachten:

```
[RO] systemd-cgtop
```

Hauptprozess anschließend mit `ps` untersuchen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,lstart,etime,stat,%cpu,%mem,rss,vsz,cmd
```

Wichtig:

- `MainPID=0` kann bedeuten, dass aktuell kein Hauptprozess vorhanden ist.
- Eine Unit kann mehrere Prozesse enthalten.
- Ein Kindprozess kann weiterlaufen, obwohl der ursprüngliche Startprozess bereits beendet wurde.
- Bei `Type=oneshot` kann eine erfolgreiche Unit aktiv sein, obwohl kein dauerhafter Prozess läuft.
- Der systemd-Status der gesamten Unit ist aussagekräftiger als die isolierte Betrachtung nur einer PID.

---

8. Prozesse unter macOS suchen und anzeigen

Prozess nach Namen oder Befehlszeile suchen:

```
[RO][SENS] pgrep -alf "<SUCHMUSTER>"
```

Exakten Prozessnamen suchen:

```
[RO] pgrep -x "<PROZESS>"
```

Bestimmte PID ausführlich anzeigen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,lstart,etime,state,%cpu,%mem,rss,vsz,command
```

Alle Prozesse ausführlich anzeigen:

```
[RO][SENS] ps -axo user,pid,ppid,lstart,etime,state,%cpu,%mem,rss,vsz,command
```

Prozesse nach aktueller CPU-Nutzung sortiert anzeigen:

```
[RO] ps -Arcwwxo pid,ppid,user,state,%cpu,%mem,rss,etime,command
```

Prozesse nach Speicherverbrauch sortiert anzeigen:

```
[RO] ps -Amcwwxo pid,ppid,user,state,%cpu,%mem,rss,etime,command
```

Interaktive Prozessübersicht nach CPU-Nutzung:

```
[RO] top -o cpu
```

Drei Messungen für eine bestimmte PID ausgeben:

```
[RO] top -l 3 -pid <PID> \
  -stats pid,command,cpu,mem,threads,state,time
```

> Bei `top -l` ist die erste CPU-Messung möglicherweise noch nicht aussagekräftig, weil für eine belastbare Prozentberechnung ein zeitlicher Vergleich benötigt wird.

---

9. launchd-Dienst einem macOS-Prozess zuordnen

Systemweiten launchd-Dienst prüfen:

```
[RO] launchctl print "system/<LABEL>"
```

Benutzerbezogenen Dienst prüfen:

```
[RO] launchctl print "gui/$(id -u)/<LABEL>"
```

In der Ausgabe sind insbesondere folgende Angaben relevant:

```
state
pid
runs
last exit code
program
arguments
```

Gefundene PID anschließend untersuchen:

```
[RO][SENS] ps -p <PID> \
  -o user,pid,ppid,lstart,etime,state,%cpu,%mem,rss,vsz,command
```

Direkte Kindprozesse einer PID anzeigen:

```
[RO][SENS] pgrep -P <PID> -a
```

Geöffnete Dateien und Netzwerkressourcen des Prozesses anzeigen:

```
[RO][SENS][PRIV] sudo lsof -nP -p <PID>
```

> Ein launchd-Job kann geladen und funktionsfähig sein, obwohl momentan keine PID angezeigt wird. Das ist bei bedarfsgesteuerten Diensten möglich.

---

10. Wichtige Prozessfelder verstehen

Feld	Bedeutung	Diagnosewert
`PID`	Prozess-ID	Identifiziert die aktuelle Instanz
`PPID`	Elternprozess-ID	Zeigt den erzeugenden Prozess
`USER`	Ausführungskonto	Wichtig für Berechtigungen
`START` beziehungsweise `LSTART`	Startzeit	Hilft beim Erkennen von Neustarts
`ETIME`	Vergangene Zeit seit Prozessstart	Zeigt kurze oder lange Laufzeit
`STAT` beziehungsweise `STATE`	Prozesszustand	Erkennt Schlafen, Warten, Stopp oder Zombie
`%CPU`	CPU-Nutzung	Hinweis auf Last oder Endlosschleife
`%MEM`	Anteil am physischen Speicher	Grober Speichervergleich
`RSS`	Aktuell residenter physischer Speicher	Praktischer Speicherindikator
`VSZ`	Virtueller Adressraum	Nicht mit tatsächlich belegtem RAM gleichsetzen
`TIME`	Verbrauchte CPU-Zeit	Kumulierte Prozessorzeit
`CMD` beziehungsweise `COMMAND`	Programm und Argumente	Identifikation und Konfigurationsprüfung
`THREADS`	Anzahl der Threads	Hinweis auf Threadwachstum
`HandleCount`	Offene Windows-Handles	Hinweis auf Ressourcenleck
`FD`	Dateideskriptor	Geöffnete Ressource unter Unix-Systemen

> Hoher virtueller Speicher bedeutet nicht automatisch, dass entsprechend viel physischer Arbeitsspeicher belegt ist.

---

11. Linux-Prozesszustände interpretieren

Der erste Buchstabe in der Spalte `STAT` beschreibt den grundlegenden Zustand.

Zustand	Bedeutung	Bewertung
`R`	Laufend oder ausführungsbereit	Bei dauerhafter hoher CPU-Nutzung untersuchen
`S`	Unterbrechbarer Schlafzustand	Häufig normal; Prozess wartet auf ein Ereignis

Zustand	Bedeutung	Bewertung
D	Nicht unterbrechbarer Wartezustand	Häufig Warten auf Ein-/Ausgabe; dauerhaft auffällig
T	Gestoppt oder durch Debugger angehalten	Ursache des Stopps prüfen
t	Während Ablaufverfolgung gestoppt	Debugger oder Tracing prüfen
Z	Zombie	Prozess ist beendet, Elternprozess hat Status noch nicht abgeholt
I	Leerlaufender Kernel-Thread	Normaler Kernelzustand
X	Beendeter Prozess	Normalerweise nur sehr kurz sichtbar

Weitere Zeichen können zusätzliche Eigenschaften angeben:

Zeichen	Bedeutung
<	Höhere Priorität
N	Niedrigere Priorität
L	Speicherseiten sind gesperrt
s	Prozess ist Sitzungsleiter
l	Mehrere Threads
+	Vordergrund-Prozessgruppe

Wichtige Bewertung:

- S ist bei Serverdiensten sehr häufig und normalerweise unauffällig.
- R ist nur zusammen mit Messdauer und CPU-Verbrauch zu bewerten.
- Ein kurzzeitiger D-Zustand kann normal sein.
- Ein dauerhaft im Zustand D verbleibender Prozess kann auf Datenträger-, Netzwerkdateisystem- oder Kernelprobleme hinweisen.
- Ein Z-Prozess verbraucht kaum Arbeitsspeicher, belegt aber weiterhin einen Eintrag in der Prozesstabelle.
- Ein Zombie wird durch den fehlerhaften oder nicht reagierenden Elternprozess bereinigt, nicht durch das Beenden des bereits beendeten Kindprozesses.

---

12. macOS-Prozesszustände interpretieren

macOS verwendet ebenfalls Zustandszeichen, deren genaue Kombination mehrere Eigenschaften enthalten kann.

Zustand	Grundbedeutung
R	Laufend oder ausführungsbereit
S	Schlafend für weniger als ungefähr 20 Sekunden
I	Länger inaktiv beziehungsweise schlafend
D	Warten auf Datenträger- oder andere nicht unterbrechbare Ein-/Ausgabe
T	Angehalten
Z	Zombie
U	Nicht unterbrechbarer Wartezustand
H	Angehaltene Ausführung
L	Prozess besitzt gesperrte Speicherseiten

Abhängig von der Ausgabe können zusätzliche Zeichen erscheinen. Deshalb sollte bei Unklarheiten die lokale Handbuchseite herangezogen werden:

```
[RO] man ps
```

> Prozesszustände dürfen nicht isoliert bewertet werden. Entscheidend ist, ob ein Zustand über einen längeren Messzeitraum bestehen bleibt und gleichzeitig eine Funktionsstörung vorliegt.

---

13. CPU-Verbrauch richtig untersuchen

Eine einzelne Momentaufnahme reicht häufig nicht aus. CPU-Nutzung sollte über mehrere Messpunkte beobachtet werden.

Beobachtung	Mögliche Erklärung
Kurzer CPU-Ausschlag	Reguläre Verarbeitung einer Anfrage
Dauerhaft hohe CPU-Nutzung	Hohe Last, Schleife, Kompression, Verschlüsselung oder fehlerhafte Verarbeitung
Niedrige CPU-Nutzung trotz Störung	Warten auf Netzwerk, Datenträger, Sperre oder Backend
Hohe gesamte CPU-Last, Dienst selbst unauffällig	Konkurrenz durch andere Prozesse
Ein Prozess über 100 Prozent unter Unix	Nutzung mehrerer CPU-Kerne möglich

Beobachtung	Mögliche Erklärung
Steigende kumulierte CPU-Zeit	Prozess verbraucht weiterhin Prozessorzeit

Windows – zwei Messpunkte vergleichen:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id, ProcessName, CPU, StartTime
```

Die Eigenschaft `CPU` enthält die bisher vom Prozess verbrauchte Prozessorzeit in Sekunden und nicht die momentane prozentuale Auslastung.

Linux – Prozess beobachten:

```
[RO] top -p <PID>
```

macOS – mehrere Messpunkte erfassen:

```
[RO] top -l 5 -s 2 -pid <PID> \
  -stats pid,command,cpu,mem,threads,state,time
```

> Ein Prozess mit niedriger CPU-Nutzung kann trotzdem blockiert sein. Warten auf Ein-/Ausgabe benötigt häufig kaum Prozessorzeit.

---

14. Speicherverbrauch richtig untersuchen

Messwert	Bedeutung
Working Set	Derzeit im physischen Speicher befindliche Speicherseiten unter Windows
Private Memory	Speicher, der dem Prozess privat zugeordnet ist
RSS	Residenter physischer Speicher unter Unix-Systemen
VSZ	Gesamter virtueller Adressraum
Swap	Auf Auslagerungsspeicher verschobene Daten
Page Fault	Zugriff auf eine nicht aktuell im Working Set befindliche Speicherseite

Windows:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                  ProcessName,
                  WorkingSet64,
                  PrivateMemorySize64,
                  VirtualMemorySize64,
                  PagedMemorySize64
```

Linux:

```
[RO] ps -p <PID> -o pid,etime,%mem,rss,vsz,stat,comm
```

macOS:

```
[RO] ps -p <PID> -o pid,etime,%mem,rss,vsz,state,command
```

Hinweise auf ein mögliches Speicherleck:

- Speicherverbrauch steigt über längere Zeit kontinuierlich,
- der Verbrauch sinkt nach abgeschlossenen Aufgaben nicht wieder,
- Auslagerungsaktivität nimmt zu,
- der Prozess wird wegen Speichermangels beendet,
- die Dienstfunktion wird mit zunehmender Laufzeit langsamer,
- ein Neustart reduziert den Speicherverbrauch nur vorübergehend.

> Ein Speicherleck lässt sich nicht durch einen einzelnen hohen Messwert beweisen. Dafür ist eine Zeitreihe unter vergleichbarer Last erforderlich.

---

15. Threads, Handles und Dateideskriptoren prüfen

Ein Prozess kann funktionsunfähig werden, obwohl er weiterhin läuft. Ursachen können erschöpfte Handles, Dateideskriptoren oder ungewöhnlich viele Threads sein.

Ressource	Windows	Linux	macOS

Threads	`Get-Process`	`ps -L`	`ps -M`
Handles	`HandleCount`	Nicht gleichbedeutend	Nicht gleichbedeutend
Dateideskriptoren	Über spezialisierte Werkzeuge	`/proc/<PID>/fd` oder `lsof`	`lsof`
Geöffnete Dateien	Sysinternals-Werkzeuge oder Prozesswerkzeuge	`lsof -p`	`lsof -p`

Windows – Thread- und Handleanzahl:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id,
                  ProcessName,
                  HandleCount,
                  @{Name="ThreadCount";Expression={$_.Threads.Count}}
```

Linux – Threads anzeigen:

```
[RO] ps -L -p <PID> -o pid,tid,psr,stat,%cpu,comm
```

Linux – geöffnete Ressourcen anzeigen:

```
[RO][SENS][PRIV] sudo lsof -nP -p <PID>
```

macOS – Threads anzeigen:

```
[RO] ps -M -p <PID>
```

macOS – geöffnete Ressourcen anzeigen:

```
[RO][SENS][PRIV] sudo lsof -nP -p <PID>
```

Verdächtig ist nicht allein eine hohe Anzahl, sondern insbesondere:

- kontinuierliches Wachstum,
- Erreichen eines konfigurierten Grenzwertes,
- Meldungen wie „too many open files“,
- fehlgeschlagene neue Verbindungen,
- steigende Threadzahl ohne entsprechende Last,

- Handles oder Dateien werden nach Aufgaben nicht freigegeben.

---

16. Wechselnde PIDs und Neustartschleifen erkennen

Eine neue PID kann bedeuten, dass ein Prozess neu gestartet wurde.

Windows – wiederholte Dienstabfrage:

```
[RO] Get-CimInstance Win32_Service -Filter "Name='<DIENSTNAME>'" |
    Select-Object Name, State, ProcessId, ExitCode
```

Linux – Neustartzähler und Hauptprozess:

```
[RO] systemctl show "<DIENST>" \
  --property=MainPID,NRestarts,ActiveEnterTimestamp,ExecMainStartTimestamp,Result
```

macOS – Anzahl der Starts und aktuelle PID:

```
[RO] launchctl print "system/<LABEL>"
```

Anzeichen einer Neustartschleife:

- PID wechselt innerhalb kurzer Zeit,
- Prozesslaufzeit bleibt sehr kurz,
- systemd-Eigenschaft `NRestarts` steigt,
- launchd-Ausgabe zeigt eine steigende Anzahl unter `runs`,
- Windows-Ereignisprotokoll enthält wiederholte Start- und Stoppmeldungen,
- Anwendung protokolliert wiederholt denselben Startfehler,
- Dienststatus wechselt zwischen `Starting`, `Running` und `Stopped`,
- ein Überwachungswerkzeug startet den Prozess fortlaufend neu.

> Ein automatisch neu gestarteter Dienst kann nach außen zeitweise erreichbar wirken. Die wechselnden PIDs und kurzen Unterbrechungen sind dann wichtige Diagnosehinweise.

---

17. Nicht reagierenden Prozess richtig bewerten

Ein Prozess kann:

- CPU-intensiv arbeiten,
- auf Datenträgerzugriff warten,
- auf eine Netzwerkantwort warten,
- auf eine Datenbanksperre warten,
- auf einen anderen Thread warten,
- durch einen Deadlock blockiert sein,
- auf Benutzereingabe warten,
- zwar reagieren, aber extrem langsam sein.

Windows-Eigenschaft `Responding`:

```
[RO] Get-Process -Id <PID> |
    Select-Object Id, ProcessName, Responding
```

Diese Eigenschaft ist hauptsächlich bei Prozessen mit grafischer Benutzeroberfläche hilfreich. Sie ist kein zuverlässiger allgemeiner Funktionstest für Serverdienste.

Bessere Bewertung eines Serverprozesses:

1. Bleibt die PID stabil?
2. Verändert sich die CPU-Zeit?
3. Lauscht der Prozess auf dem erwarteten Port?
4. Nimmt er neue Verbindungen an?
5. Reagiert das Anwendungsprotokoll?
6. Gibt es blockierte Ein-/Ausgabe?
7. Steigt die Anzahl offener Verbindungen oder Dateien?
8. Zeigen Protokolle Zeitüberschreitungen oder Sperren?
9. Funktionieren abhängige Systeme?
10. Reagiert der Prozess auf einen kontrollierten Diagnosetest?

> „Läuft“ und „reagiert“ sind unterschiedliche Zustände.

---

18. Prozess beenden – nur als kontrollierte Maßnahme

Die folgenden Befehle verändern den Systemzustand und können Datenverlust oder Dienstunterbrechungen verursachen.

Aufgabe	Windows	Linux	macOS
Reguläre Beendigung anfordern	`[CHANGE][DISRUPT] Stop-Process -Id <PID>`	`[CHANGE][DISRUPT] kill -TERM <PID>`	`[CHANGE][DISRUPT] kill -TERM <PID>`
Beendigung erzwingen	`[CHANGE][DISRUPT][PRIV] Stop-Process -Id <PID> -Force`	`[CHANGE][DISRUPT][PRIV] sudo kill -KILL <PID>`	`[CHANGE][DISRUPT][PRIV] sudo kill -KILL <PID>`

Vorher prüfen:

- [ ] PID unmittelbar vor der Maßnahme erneut verifiziert
- [ ] Prozessname und Dienstzuordnung bestätigt
- [ ] Aktive Benutzer und Verbindungen geprüft
- [ ] Laufende Schreibvorgänge und Transaktionen berücksichtigt
- [ ] Prozess- und Dienstprotokolle gesichert
- [ ] Abhängige Dienste geprüft
- [ ] Cluster- oder Failover-Verhalten berücksichtigt
- [ ] Automatische Neustartregel geprüft
- [ ] Betriebliche Freigabe vorhanden
- [ ] Funktionstest nach der Maßnahme vorbereitet

Wichtige Unterschiede:

- `SIGTERM` gibt einem Unix-Prozess die Möglichkeit, kontrolliert zu beenden.
- `SIGKILL` kann nicht vom Prozess behandelt oder abgefangen werden.
- `Stop-Process -Force` erzwingt die Beendigung unter Windows.
- Wird nur der Prozess beendet, kann die Dienstverwaltung ihn automatisch erneut starten.
- Bei verwalteten Diensten sollte grundsätzlich die jeweilige Dienstverwaltung verwendet werden.
- Eine erzwungene Beendigung ist kein Ersatz für eine Ursachenanalyse.

> Niemals eine alte, zuvor notierte PID ungeprüft verwenden. Die PID könnte inzwischen einem anderen Prozess gehören.

---

19. Typische Fehlinterpretationen

Fehlinterpretation	Richtige Bewertung
„Die PID existiert, also funktioniert der Dienst.“	Nur die Prozessinstanz wurde nachgewiesen
„Der Prozess verwendet wenig CPU, also ist er gesund.“	Er kann auf Ein-/Ausgabe oder eine Sperre warten
„Hohe CPU-Nutzung ist immer ein Fehler.“	Sie kann durch reguläre Last entstehen
„Hoher virtueller Speicher ist vollständig belegter RAM.“	Virtueller Adressraum und residenter Speicher unterscheiden sich
„Ein Zombie verbraucht sehr viel Arbeitsspeicher.“	Er belegt hauptsächlich einen Prozesslisteneintrag
„Der Zombie muss mit `kill` beendet werden.“	Der Prozess ist bereits beendet; der Elternprozess muss seinen Status abholen
„Eine neue PID ist unproblematisch.“	Sie kann einen Absturz und automatischen Neustart anzeigen
„Der Prozessname identifiziert den Dienst eindeutig.“	Mehrere Dienste können denselben Prozessnamen verwenden
„`Responding=True` beweist die Serverfunktion.“	Es ist kein vollständiger Anwendungstest
„SIGKILL ist schneller und deshalb besser.“	Es verhindert eine kontrollierte Bereinigung und erhöht das Risiko

---

20. Checkliste zur Prozessanalyse

[ ] Dienst dem richtigen Prozess zugeordnet
[ ] PID unmittelbar aktuell ermittelt
[ ] Prozessname geprüft
[ ] Elternprozess und Kindprozesse geprüft
[ ] Ausführungskonto geprüft
[ ] Programmpfad und Befehlszeile geprüft
[ ] Startzeit und bisherige Laufzeit geprüft
[ ] Prozesszustand geprüft
[ ] CPU über mehrere Messpunkte beobachtet
[ ] Speicherverbrauch über einen Zeitraum bewertet
[ ] Threadanzahl geprüft
[ ] Handles beziehungsweise Dateideskriptoren geprüft
[ ] Geöffnete Dateien und Sockets bei Bedarf geprüft
[ ] Wechselnde PID ausgeschlossen oder dokumentiert
[ ] Automatische Neustarts geprüft
[ ] Exitcode und Protokolle gesichert

```
[ ] Noch keine ungeprüfte Prozessbeendigung durchgeführt
[ ] Nächsten Diagnoseschritt festgelegt
```

Bewertung des Ergebnisses

Ergebnis	Nächster Schritt
Kein Prozess vorhanden, Dienst sollte laufen	Dienststartfehler und Protokolle prüfen
Prozess wird fortlaufend neu gestartet	Exitcode, Neustartregel und Startprotokoll untersuchen
Prozess läuft stabil	Listener, Ports und Anwendungsfunktion prüfen
Prozess verwendet dauerhaft viel CPU	Threads, Anfragen, Schleifen und Last untersuchen
Speicherverbrauch steigt kontinuierlich	Zeitreihe, Grenzwerte und mögliche Speicherlecks untersuchen
Prozess verbleibt in einem Wartezustand	Datenträger, Netzwerk, Sperren und Backends prüfen
Zombieprozesse sammeln sich	Elternprozess und dessen Fehlerbehandlung untersuchen
Handle- oder Dateideskriptoranzahl steigt	Ressourcenleck und Grenzwerte untersuchen
Falsches Ausführungskonto	Dienstkonfiguration und Berechtigungen prüfen
Unerwarteter Elternprozess	Startweg, Überwachung, Wrapper oder Sicherheitsvorfall untersuchen

---

Merksatz

> **Der Dienststatus zeigt die Sicht der Dienstverwaltung. Die Prozessanalyse zeigt, was die ausführende Instanz tatsächlich tut – oder worauf sie wartet.**

Weiterführende Quellen

- Microsoft Learn – Get-Process
- Microsoft Learn – Win32_Process
- Microsoft Learn – Win32_Service
- Microsoft Learn – tasklist
- Linux-Handbuch – ps
- Linux-Handbuch – proc
- Linux-Handbuch – pgrep

- systemd – systemctl
- systemd – systemd-cgls
- Apple – ps-Handbuchseite
- Apple – pgrep-Handbuchseite
- Apple – top-Handbuchseite
- Apple – Aktivitätsanzeige – Benutzerhandbuch