4.6 System- und Dienstprotokolle auswerten

System- und Dienstprotokolle dokumentieren Ereignisse, die während des Starts, Betriebs und Beendens eines Dienstes auftreten. Sie können Hinweise auf Konfigurationsfehler, fehlende Berechtigungen, nicht erreichbare Abhängigkeiten, Ressourcenmangel oder Programmabstürze liefern.

> **Grundsatz:**
> Protokolle werden nicht wahllos nach dem Wort „Fehler“ durchsucht. Zuerst werden Zeitraum, betroffener Dienst, Server und auslösende Aktion festgelegt.

Ziele dieser Seite

Nach dieser Seite sollst du:

- ein geeignetes Zeitfenster für die Protokollanalyse bestimmen können,
- Windows-Ereignisprotokolle, systemd-Journal und macOS Unified Logging auswerten können,
- nach Dienst, Quelle, Prozess, Ereignis-ID und Priorität filtern können,
- System- und Anwendungsprotokolle zeitlich miteinander verbinden können,
- Ursachen von Folgefehlern unterscheiden können,
- Live-Protokolle kontrolliert beobachten können,
- relevante Protokolle sichern können,
- sensible Inhalte vor einer Weitergabe erkennen können,
- Grenzen und Aufbewahrungsregeln von Protokollen berücksichtigen können.

---

1. Beobachtung, Protokollhinweis und Ursache unterscheiden

Ebene	Beispiel
Benutzerbeobachtung	Anmeldung schlägt um 09:15 Uhr fehl
Anwendungsprotokoll	Datenbankverbindung um 09:15:02 Uhr fehlgeschlagen
Systemprotokoll	Netzwerkschnittstelle um 09:14:58 Uhr getrennt
Backendprotokoll	Keine Anfrage vom Anwendungsserver eingegangen
Ursache	Netzwerkschnittstelle verlor aufgrund eines Treiberfehlers die Verbindung
Folgefehler	Anwendung meldete anschließend einen Datenbankfehler

Eine Protokollmeldung kann sein:

- direkte Ursache,
- Folge einer anderen Störung,
- Warnung ohne Bezug zum aktuellen Problem,
- erwarteter Betriebszustand,
- Wiederholungsversuch,
- Ergebnis einer automatischen Wiederherstellung,
- Meldung eines vorgeschalteten oder abhängigen Systems.

> Der erste sichtbare Fehler in einer Anwendung ist nicht zwingend das zeitlich erste technische Ereignis.

---

2. Vor der Suche ein genaues Zeitfenster festlegen

Folgende Zeitangaben müssen möglichst genau bestimmt werden:

Information	Beispiel
Beginn des Fehlers	2026-07-31T09:15:00+02:00
Ende beziehungsweise letzter Fehler	2026-07-31T09:22:00+02:00
Zeitzone	Europe/Berlin, UTC+02:00
Auslösende Aktion	Anmeldung abgesendet
Clientzeit	09:15:00
Serverzeit	09:15:02
Backendzeit	07:15:02 UTC
Vergleichszeitpunkt	Letzte erfolgreiche Anmeldung um 09:13 Uhr

Empfohlener Suchbereich:

```
Einige Minuten vor dem ersten sichtbaren Fehler
bis
einige Minuten nach dem letzten beobachteten Fehler
```

Prüffragen:

- Verwenden alle Systeme dieselbe Zeitzone?

- Protokolliert eine Anwendung in UTC?
- Sind die Systemuhren synchronisiert?
- Ist der Clientzeitstempel nur auf Sekunden genau?
- Gab es unmittelbar vorher einen Neustart oder eine Änderung?
- Ist das Problem dauerhaft oder nur kurzzeitig aufgetreten?

---

3. Protokollanalyse mit einer klaren Hypothese beginnen

Ungeeignete Suche:

> „Ich suche in allen Protokollen nach Fehlern.“

Geeignete Suche:

> „Ich untersuche zwischen 09:10 und 09:20 Uhr die Ereignisse des Anwendungsdienstes, des Service Control Managers und der Datenbankverbindung.“

Eine gute Protokollsuche verwendet möglichst mehrere Filter:

- Zeitfenster,
- Server,
- Dienstname,
- Prozessname,
- Prozess-ID,
- Ereignisquelle,
- Ereignis-ID,
- Priorität,
- Transaktions- oder Korrelations-ID,
- Benutzer oder Sitzung,
- Zielserver,
- Fehlercode.

Empfohlene Reihenfolge:

```
1. Zeitpunkt festlegen
2. Hauptdienst filtern
```

3. Erste relevante Fehlermeldung bestimmen
4. Einige Ereignisse davor lesen
5. Abhängige Systeme prüfen
6. Fehlercode in offizieller Dokumentation nachschlagen
7. Ereignisse zeitlich zusammenführen
8. Hypothese mit einem gezielten Test überprüfen

---

4. Protokollquellen der Betriebssysteme vergleichen

Bereich	Windows	Linux mit systemd	macOS
Zentrale Oberfläche	Ereignisanzeige	`journalctl`	Konsole
Zentrales Befehlswerkzeug	`Get-WinEvent`	`journalctl`	`log`
Systemereignisse	Protokoll `System`	System-Journal	Unified Logging
Anwendungsereignisse	Protokoll `Application` oder eigenes Protokoll	Journal oder Anwendungsdatei	Unified Logging oder Anwendungsdatei
Dienstverwaltung	Service Control Manager	systemd	launchd
Kernelmeldungen	Protokoll `System`	`journalctl -k`	`log show` und Diagnoseberichte
Absturzberichte	Windows Error Reporting	Coredump- oder Anwendungsmechanismus	Diagnoseberichte
Dateibasierte Logs	Anwendungsspezifisch	Häufig unter `/var/log` oder Produktpfad	Anwendungsspezifisch

> Nicht jede Anwendung schreibt in das zentrale Betriebssystemprotokoll. Herstellerdokumentation und effektive Dienstkonfiguration müssen zusätzlich geprüft werden.

---

5. Verfügbare Windows-Ereignisprotokolle ermitteln

Alle registrierten Protokolle auflisten:

```
[RO] Get-WinEvent -ListLog *
```

Nur aktivierte Protokolle anzeigen:

```
[RO] Get-WinEvent -ListLog * |
    Where-Object IsEnabled |
    Select-Object LogName, RecordCount, FileSize, MaximumSizeInBytes |
    Sort-Object LogName
```

Bestimmtes Protokoll untersuchen:

```
[RO] Get-WinEvent -ListLog "System"
```

Registrierte Ereignisanbieter suchen:

```
[RO] Get-WinEvent -ListProvider "*<SUCHBEGRIFF>*"
```

Typische Windows-Protokolle:

Protokoll	Typischer Inhalt
`System`	Dienste, Treiber, Netzwerk und Betriebssystem
`Application`	Anwendungen und Laufzeitumgebungen
`Security`	Sicherheitsereignisse und Überwachung
`Setup`	Installation und Systemeinrichtung
`ForwardedEvents`	Weitergeleitete Ereignisse
Anwendungs- und Dienstprotokolle	Komponenten- oder produktspezifische Ereignisse

> Der Zugriff auf das Sicherheitsprotokoll und bestimmte administrative Protokolle erfordert erhöhte Berechtigungen.

---

6. Letzte Windows-Ereignisse anzeigen

Letzte 50 Systemereignisse anzeigen:

```
[RO] Get-WinEvent -LogName "System" -MaxEvents 50 |
    Select-Object TimeCreated,
                  Id,
                  LevelDisplayName,
                  ProviderName,
                  Message
```

Letzte 50 Anwendungsereignisse anzeigen:

```
[RO] Get-WinEvent -LogName "Application" -MaxEvents 50 |
    Select-Object TimeCreated,
                  Id,
                  LevelDisplayName,
                  ProviderName,
                  Message
```

Neueste Ereignisse zuerst tabellarisch anzeigen:

```
[RO] Get-WinEvent -LogName "System" -MaxEvents 100 |
    Format-Table TimeCreated,
                 Id,
                 LevelDisplayName,
                 ProviderName,
                 Message -Wrap
```

> `Format-Table` sollte hauptsächlich für die Anzeige verwendet werden. Für Export oder Weiterverarbeitung bleiben die ursprünglichen Ereignisobjekte geeigneter.

---

7. Windows-Ereignisse nach Zeitfenster filtern

Ereignisse der letzten Stunde:

```
[RO] $startTime = (Get-Date).AddHours(-1)

Get-WinEvent -FilterHashtable @{
    LogName   = "System"
    StartTime = $startTime
} |
    Select-Object TimeCreated,
                  Id,
                  LevelDisplayName,
                  ProviderName,
                  Message
```

Festes Zeitfenster verwenden:

```
[RO] $startTime = Get-Date "2026-07-31 09:10:00"
$endTime   = Get-Date "2026-07-31 09:20:00"

Get-WinEvent -FilterHashtable @{
    LogName   = "System"
    StartTime = $startTime
    EndTime   = $endTime
} |
    Select-Object TimeCreated,
                  Id,
                  LevelDisplayName,
                  ProviderName,
                  Message
```

System- und Anwendungsprotokoll gemeinsam durchsuchen:

```
[RO] Get-WinEvent -FilterHashtable @{
    LogName   = "System", "Application"
    StartTime = $startTime
    EndTime   = $endTime
} |
    Sort-Object TimeCreated |
    Select-Object TimeCreated,
```

```
    LogName,
    Id,
    LevelDisplayName,
    ProviderName,
    Message
```

> Die Uhrzeitangaben werden im Kontext des ausführenden Systems interpretiert. Zeitzone und Uhrzeitsynchronisation müssen deshalb vor der Korrelation geklärt sein.

---

8. Windows-Ereignisse nach Quelle, ID und Schweregrad filtern

Nach Ereignisanbieter filtern:

```
[RO] Get-WinEvent -FilterHashtable @{
    LogName      = "System"
    ProviderName = "Service Control Manager"
    StartTime    = (Get-Date).AddHours(-2)
}
```

Nach bestimmten Ereignis-IDs filtern:

```
[RO] Get-WinEvent -FilterHashtable @{
    LogName      = "System"
    ProviderName = "Service Control Manager"
    Id           = 7000, 7001, 7031, 7034
    StartTime    = (Get-Date).AddHours(-24)
} |
    Select-Object TimeCreated, Id, LevelDisplayName, Message
```

Nur kritische Ereignisse und Fehler anzeigen:

```
[RO] Get-WinEvent -FilterHashtable @{
    LogName   = "System"
```

```
    Level     = 1, 2
    StartTime = (Get-Date).AddHours(-2)
}
```

Übliche Windows-Ereignisebenen:

Zahlenwert	Ebene
1	Kritisch
2	Fehler
3	Warnung
4	Information
5	Ausführlich

> Ereignis-IDs sind nur zusammen mit Anbieter und Protokoll eindeutig zu bewerten. Dieselbe ID kann bei unterschiedlichen Anbietern eine andere Bedeutung besitzen.

---

9. Typische Windows-Dienstereignisse einordnen

Häufig relevante Ereignisse des Service Control Managers sind:

Ereignis-ID	Typische Bedeutung
7000	Dienst konnte nicht gestartet werden
7001	Abhängiger Dienst oder Abhängigkeitsgruppe konnte nicht gestartet werden
7009	Zeitüberschreitung beim Warten auf eine Dienstverbindung
7011	Zeitüberschreitung bei einer Diensttransaktion
7023	Dienst wurde mit einem Fehler beendet
7024	Dienst wurde mit einem dienstspezifischen Fehler beendet
7031	Dienst wurde unerwartet beendet; Wiederherstellungsaktion kann folgen
7034	Dienst wurde unerwartet beendet

Ereignis-ID	Typische Bedeutung
7035	Steuerbefehl wurde an einen Dienst gesendet
7036	Dienst wechselte in einen anderen Zustand
7040	Startart wurde geändert
7045	Ein Dienst wurde im System installiert

Wichtig:

- Die konkrete Nachricht muss immer mitgelesen werden.
- Die Bedeutung muss bei Bedarf anhand der Microsoft-Dokumentation und des betroffenen Produkts geprüft werden.
- Ein Service-Control-Manager-Ereignis kann nur die Auswirkung eines internen Anwendungsfehlers melden.
- Für die eigentliche Ursache muss häufig zusätzlich das Anwendungsprotokoll untersucht werden.

---

10. Windows-Ereignisse nach Nachrichtentext durchsuchen

Nach einem Dienstnamen innerhalb eines begrenzten Zeitraums suchen:

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName   = "System", "Application"
    StartTime = (Get-Date).AddHours(-2)
} |
    Where-Object Message -Match "<DIENSTNAME>" |
    Select-Object TimeCreated,
                  LogName,
                  Id,
                  LevelDisplayName,
                  ProviderName,
                  Message
```

Nach einem Fehlercode suchen:

```
[RO][SENS] Get-WinEvent -FilterHashtable @{
    LogName   = "System", "Application"
    StartTime = (Get-Date).AddHours(-24)
```

```
} |
    Where-Object Message -Match "<FEHLERCODE>"
```

> Textfilter mit `Where-Object` werden erst nach dem Lesen der Ereignisse angewendet und können bei großen Protokollen langsam sein. Zeitfenster, Protokoll und Anbieter sollten deshalb bereits serverseitig mit `FilterHashtable` eingeschränkt werden.

---

11. Windows-Ereignisse mit wevtutil prüfen und sichern

Informationen zu einem Protokoll anzeigen:

```
[RO] wevtutil get-log "System"
```

Kurzform:

```
[RO] wevtutil gl "System"
```

Die letzten Ereignisse eines Protokolls abfragen:

```
[RO] wevtutil query-events "System" /count:20 /reverse-direction:true /format:text
```

Kurzform:

```
[RO] wevtutil qe "System" /c:20 /rd:true /f:text
```

Ein vollständiges Ereignisprotokoll exportieren:

```
[RO][FILE][SENS][PRIV] wevtutil export-log `
    "System" `
    "<AUSGABEDATEI>.evtx"
```

Kurzform:

```
[RO][FILE][SENS][PRIV] wevtutil epl "System" "<AUSGABEDATEI>.evtx"
```

> Eine EVTX-Datei kann Benutzernamen, Rechnernamen, interne Pfade, IP-Adressen und sicherheitsrelevante Ereignisse enthalten. Sie darf nur kontrolliert weitergegeben werden.

---

12. Linux-Journal eines Dienstes anzeigen

Gesamtes Journal einer systemd-Unit anzeigen:

```
[RO][SENS][PRIV] sudo journalctl -u "<DIENST>" --no-pager
```

Neueste Einträge zuerst anzeigen:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --reverse \
  --no-pager
```

Letzte 100 Einträge anzeigen:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  -n 100 \
  --no-pager
```

Nur Ereignisse des aktuellen Systemstarts:

```
[RO][SENS][PRIV] sudo journalctl \
  -b \
  -u "<DIENST>" \
  --no-pager
```

Ereignisse des vorherigen Systemstarts:

```
[RO][SENS][PRIV] sudo journalctl \
  -b -1 \
  -u "<DIENST>" \
  --no-pager
```

> Ob Ereignisse früherer Systemstarts verfügbar sind, hängt von der Journal-Konfiguration und Aufbewahrung ab.

---

13. Linux-Journal nach Zeitfenster filtern

Ereignisse seit einer festen Uhrzeit:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --since "2026-07-31 09:10:00" \
  --no-pager
```

Festes Start- und Endzeitfenster:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --since "2026-07-31 09:10:00" \
  --until "2026-07-31 09:20:00" \
  --no-pager
```

Relative Zeitangabe:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --since "1 hour ago" \
  --no-pager
```

Alle Ereignisse des heutigen Tages:

```
[RO][SENS][PRIV] sudo journalctl --since today --no-pager
```

Zeitstempel mit ISO-Datum ausgeben:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  --since "1 hour ago" \
  --output=short-iso \
  --no-pager
```

---

14. Linux-Journal nach Priorität filtern

Übliche syslog-Prioritäten:

Wert	Name	Bedeutung
0	emerg	System unbenutzbar
1	alert	Sofortige Maßnahme erforderlich
2	crit	Kritischer Zustand
3	err	Fehler
4	warning	Warnung
5	notice	Bedeutender normaler Zustand
6	info	Information
7	debug	Debugmeldung

Fehler und schwerwiegendere Meldungen eines Dienstes:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  -p err \
  --since "1 hour ago" \
  --no-pager
```

Warnungen und schwerwiegendere Meldungen:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  -p warning \
  --since "1 hour ago" \
  --no-pager
```

Nur einen bestimmten Prioritätsbereich verwenden:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  -p warning..err \
  --since "1 hour ago" \
  --no-pager
```

> Nicht jede Anwendung verwendet Prioritäten korrekt. Eine relevante Fehlermeldung kann deshalb als `info` oder ohne geeignete Priorität protokolliert worden sein.

---

15. Linux-Journal nach Prozess oder Kennung filtern

Nach Prozess-ID filtern:

```
[RO][SENS][PRIV] sudo journalctl _PID=<PID> --no-pager
```

Nach ausführbarer Datei filtern:

```
[RO][SENS][PRIV] sudo journalctl \
  _EXE="<VOLLSTÄNDIGER_PROGRAMMPFAD>" \
  --no-pager
```

Nach syslog-Kennung filtern:

```
[RO][SENS][PRIV] sudo journalctl \
  -t "<KENNUNG>" \
  --no-pager
```

Felder eines Ereignisses ausführlich anzeigen:

```
[RO][SENS][PRIV] sudo journalctl \
  -u "<DIENST>" \
  -n 20 \
  -o verbose \
  --no-pager
```

Mögliche strukturierte Felder:

```
_SYSTEMD_UNIT
_PID
_UID
_GID
_EXE
_COMM
_HOSTNAME
SYSLOG_IDENTIFIER
PRIORITY
MESSAGE
```

> PID-Filter sind nur für die Lebensdauer der jeweiligen Prozessinstanz geeignet. Nach einem Neustart besitzt der Dienst möglicherweise eine neue PID.

---

16. Linux-Kernel- und Bootmeldungen prüfen

Kernelmeldungen des aktuellen Systemstarts:

```
[RO][SENS][PRIV] sudo journalctl -k -b --no-pager
```

Kernelwarnungen und schwerwiegendere Meldungen:

```
[RO][SENS][PRIV] sudo journalctl \
  -k \
  -b \
  -p warning \
  --no-pager
```

Alle Meldungen des aktuellen Systemstarts:

```
[RO][SENS][PRIV] sudo journalctl -b --no-pager
```

Verfügbare Systemstarts anzeigen:

```
[RO] journalctl --list-boots
```

Relevante Kernelhinweise können sein:

- Datenträger- und Dateisystemfehler,
- Netzwerkadapterfehler,
- Speichermangel,
- Prozessbeendigung durch den Out-of-Memory-Mechanismus,
- Treiberprobleme,
- schreibgeschützte Dateisysteme,
- blockierte Aufgaben,
- Hardwarefehler.

---

17. Linux-Journal live beobachten

Neue Ereignisse eines Dienstes live anzeigen:

```
[TEST][SENS][PRIV] sudo journalctl \
  -f \
  -u "<DIENST>"
```

Mit bestehenden letzten Zeilen beginnen:

```
[TEST][SENS][PRIV] sudo journalctl \
  -n 50 \
  -f \
  -u "<DIENST>"
```

Nur Warnungen und schwerwiegendere Meldungen verfolgen:

```
[TEST][SENS][PRIV] sudo journalctl \
  -f \
  -u "<DIENST>" \
  -p warning
```

Vorgehen beim kontrollierten Live-Test:

```
1. Live-Ansicht starten
2. Genauen Startzeitpunkt notieren
3. Eine einzelne Testaktion ausführen
4. Neue Ereignisse beobachten
5. Testaktion und Ereignisse zeitlich zuordnen
6. Live-Ansicht mit Strg+C beenden
7. Relevante Meldungen dokumentieren
```

> Live-Logging kann große Mengen ausgeben. Der Filter sollte vor dem Test möglichst eng gesetzt werden.

---

18. Linux-Journalgröße und Aufbewahrung prüfen

Belegten Speicherplatz des Journals anzeigen:

```
[RO] journalctl --disk-usage
```

Verfügbare Systemstarts anzeigen:

```
[RO] journalctl --list-boots
```

Journal-Konfiguration lesen:

```
[RO][FILE] systemd-analyze cat-config systemd/journald.conf
```

Wichtige Konfigurationsbereiche können sein:

```
Storage
SystemMaxUse
RuntimeMaxUse
MaxRetentionSec
MaxFileSec
RateLimitIntervalSec
RateLimitBurst
```

Mögliche Gründe für fehlende alte Einträge:

- Journal wird nur flüchtig gespeichert,
- Größenbegrenzung wurde erreicht,
- Aufbewahrungszeit ist abgelaufen,
- System wurde neu gestartet,
- Protokollrotation hat alte Daten entfernt,
- Rate-Limiting hat Meldungen unterdrückt,
- Anwendung schrieb in eine andere Datei,
- Dienst startete in einem Container mit eigener Protokollierung.

> Aufbewahrungseinstellungen dürfen nicht während einer Störung ungeprüft verändert werden. Zuerst muss der aktuelle Zustand dokumentiert werden.

---

19. Textbasierte Linux-Anwendungsprotokolle prüfen

Der tatsächliche Protokollpfad muss aus Herstellerdokumentation oder effektiver Konfiguration ermittelt werden.

Letzte Zeilen einer bekannten Protokolldatei:

```
[RO][FILE][SENS][PRIV] sudo tail -n 100 "<PROTOKOLLDATEI>"
```

Neue Einträge live verfolgen:

```
[TEST][FILE][SENS][PRIV] sudo tail -F "<PROTOKOLLDATEI>"
```

Nach einem festen Text suchen:

```
[RO][FILE][SENS][PRIV] sudo grep \
  -n \
  -i \
  -- "<SUCHTEXT>" \
  "<PROTOKOLLDATEI>"
```

Komprimierte rotierte Protokolle durchsuchen:

```
[RO][FILE][SENS][PRIV] sudo zgrep \
  -n \
  -i \
  -- "<SUCHTEXT>" \
  "<PROTOKOLLDATEI>.gz"
```

Datei kontrolliert mit Pager öffnen:

```
[RO][FILE][SENS][PRIV] sudo less "<PROTOKOLLDATEI>"
```

Nützliche Tasten in `less`:

Taste	Funktion
`/Text`	Vorwärts suchen
`?Text`	Rückwärts suchen
`n`	Nächster Treffer
`N`	Vorheriger Treffer
`G`	Dateiende
`g`	Dateianfang

Taste	Funktion
q	Beenden

> Keine angenommenen Standardpfade verwenden. Container, Pakete und Hersteller können unterschiedliche Protokollziele konfigurieren.

---

20. macOS Unified Logging nach Zeitraum auswerten

Meldungen der letzten Stunde anzeigen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --style compact \
  --no-pager
```

Meldungen seit dem letzten Systemstart:

```
[RO][SENS][PRIV] sudo log show \
  --last boot \
  --style compact \
  --no-pager
```

Festes Zeitfenster verwenden:

```
[RO][SENS][PRIV] sudo log show \
  --start "2026-07-31 09:10:00" \
  --end "2026-07-31 09:20:00" \
  --style compact \
  --no-pager
```

Lokale Zeitzone ausdrücklich verwenden:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
```

```
  --timezone local \
  --style compact \
  --no-pager
```

> Standardmäßig zeigt `log show` nicht zwingend alle Info- und Debugmeldungen. Diese Ebenen werden bei Bedarf ausdrücklich aktiviert.

---

21. macOS-Protokolle nach Prozess filtern

Nach Prozessnamen filtern:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'process == "<PROZESS>"' \
  --style compact \
  --no-pager
```

Nach Prozess-ID filtern:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'processIdentifier == <PID>' \
  --style compact \
  --no-pager
```

Info-Meldungen einschließen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --info \
  --predicate 'process == "<PROZESS>"' \
  --style compact \
  --no-pager
```

Info- und Debugmeldungen einschließen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --info \
  --debug \
  --predicate 'process == "<PROZESS>"' \
  --style compact \
  --no-pager
```

> Die Erhöhung des sichtbaren Umfangs kann sehr große Ausgaben erzeugen. Zuerst sollte mit engem Zeitfenster und Prozessfilter gearbeitet werden.

---

22. macOS-Protokolle nach Subsystem und Schweregrad filtern

Nach bekanntem Subsystem filtern:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'subsystem == "<SUBSYSTEM>"' \
  --style compact \
  --no-pager
```

Fehler und Faults eines Prozesses anzeigen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'process == "<PROZESS>" AND (messageType == error OR messageType == fault)' \
  --style compact \
  --no-pager
```

Nach Text innerhalb der Meldung suchen:

```
[RO][SENS][PRIV] sudo log show \
  --last 1h \
  --predicate 'eventMessage CONTAINS[c] "<SUCHTEXT>"' \
  --style compact \
  --no-pager
```

Wichtige Filterfelder:

Feld	Bedeutung
`process`	Name des erzeugenden Prozesses
`processIdentifier`	PID
`subsystem`	Logisches Subsystem
`category`	Kategorie innerhalb des Subsystems
`messageType`	Meldungstyp
`eventMessage`	Meldungstext
`sender`	Bibliothek oder ausführende Komponente
`processImagePath`	Programmpfad

---

23. macOS-Protokolle live beobachten

Meldungen eines Prozesses live verfolgen:

```
[TEST][SENS][PRIV] sudo log stream \
  --predicate 'process == "<PROZESS>"' \
  --style compact
```

Fehler und Faults live verfolgen:

```
[TEST][SENS][PRIV] sudo log stream \
  --predicate 'process == "<PROZESS>" AND (messageType == error OR messageType == fault)' \
  --style compact
```

Live-Ausgabe zeitlich begrenzen:

```
[TEST][SENS][PRIV] sudo log stream \
  --timeout 5m \
  --predicate 'process == "<PROZESS>"' \
  --style compact
```

Info-Ebene einbeziehen:

```
[TEST][SENS][PRIV] sudo log stream \
  --level info \
  --predicate 'process == "<PROZESS>"' \
  --style compact
```

> Der Befehl wird ohne Zeitbegrenzung mit `Strg+C` beendet. Eine zeitliche Begrenzung verhindert versehentlich lange laufende Mitschnitte.

---

24. macOS-Protokollarchiv sichern

Protokolle der letzten Stunde in ein Logarchiv sammeln:

```
[RO][FILE][SENS][PRIV] sudo log collect \
  --last 1h \
  --output "<AUSGABEDATEI>.logarchive"
```

Protokolle ab einem festen Zeitpunkt sammeln:

```
[RO][FILE][SENS][PRIV] sudo log collect \
  --start "2026-07-31 09:10:00" \
  --output "<AUSGABEDATEI>.logarchive"
```

Ein `.logarchive` kann später mit der App „Konsole“ oder mit `log show` ausgewertet werden.

Archiv über die Kommandozeile lesen:

```
[RO][FILE][SENS] log show \
  --archive "<AUSGABEDATEI>.logarchive" \
  --style compact \
  --no-pager
```

> Ein vollständiges macOS-Logarchiv kann umfangreiche und sensible Systeminformationen enthalten. Vor einer externen Weitergabe müssen Zweck, Empfänger und Datenschutz geprüft werden.

---

25. Anwendungsprotokolle unter Windows und macOS berücksichtigen

Nicht jede Anwendung verwendet ausschließlich das zentrale Systemprotokoll.

Mögliche Protokollziele:

- Windows-Anwendungsprotokoll,
- eigenes Windows-Ereignisprotokoll,
- macOS Unified Logging,
- Textdatei,
- JSON-Datei,
- Datenbank,
- Container-Standardausgabe,
- zentraler Syslog-Server,
- SIEM- oder Monitoringplattform,
- herstellerspezifisches Diagnosepaket.

Protokollpfad korrekt ermitteln über:

1. Herstellerdokumentation,
2. wirksame Dienstkonfiguration,
3. Startparameter,
4. Umgebungsvariablen,
5. Dienstkonto,
6. Containerdefinition,
7. Protokollierungs- oder Loggingabschnitt der Anwendung.

>

> Keine Verzeichnisse oder Dateinamen als gegeben annehmen. Der wirksame Pfad kann durch Installation, Paketierung oder lokale Konfiguration verändert worden sein.

---

26. Containerprotokolle berücksichtigen

Bei containerisierten Diensten können Protokolle an mehreren Stellen entstehen:

```
Hostbetriebssystem
  → Container-Runtime
  → Container
  → Prozess im Container
  → Anwendung
  → externes zentrales Logging
```

Mögliche Protokollquellen:

- Runtime-Ereignisse auf dem Host,
- Standardausgabe und Standardfehler des Containers,
- an ein Volume gebundene Anwendungsprotokolle,
- Reverse-Proxy-Protokolle,
- Datenbankprotokolle,
- Orchestrator-Ereignisse,
- Healthcheck-Ausgaben.

Prüffragen:

- Wurde der Container neu erstellt?
- Ist die Container-ID gewechselt?
- Wurden alte Containerprotokolle entfernt?
- Schreibt die Anwendung in die Standardausgabe oder in eine Datei?
- Ist das Logverzeichnis dauerhaft eingebunden?
- Gibt es Größenbegrenzung und Rotation?
- Werden Protokolle an einen zentralen Dienst weitergeleitet?
- Enthält die Ausgabe Secrets oder personenbezogene Daten?

>

> Die konkrete Runtime wird in dieser Seite nicht vorausgesetzt. Die passenden Befehle müssen anhand der tatsächlich eingesetzten Containerplattform ausgewählt werden.

---

27. Protokolle mehrerer Systeme zeitlich korrelieren

Beispiel:

Zeit	System	Ereignis
09:14:58	Datenbankserver	Datenträger meldet Schreibfehler
09:15:00	Datenbankserver	Datenbank beendet Schreibtransaktion mit Fehler
09:15:02	Anwendungsserver	Datenbankverbindung wird zurückgesetzt
09:15:03	Webserver	Backend antwortet nicht
09:15:04	Client	HTTP 503 wird angezeigt

Wahrscheinliche Ereigniskette:

```
Datenträgerfehler
  → Datenbankfehler
  → Verbindung des Anwendungsservers bricht ab
  → Webserver erhält keine gültige Backendantwort
  → Benutzer sieht HTTP 503
```

Vorgehen:

1. Alle Zeitstempel in dieselbe Zeitzone überführen.
2. Erstes technisches Ereignis bestimmen.
3. Vorhergehende Warnungen berücksichtigen.
4. Fehler auf Quell- und Zielsystem vergleichen.
5. Verbindungs- oder Korrelations-IDs verwenden.
6. Ursache und Folge in einer Zeitleiste dokumentieren.
7. Hypothese mit technischen Tests bestätigen.

---

28. Fehlercodes richtig auswerten

Ein Fehlercode sollte zusammen mit folgendem Kontext dokumentiert werden:

```
Betriebssystem:
Produkt:
Komponente:
Version:
Protokoll:
Ereignisanbieter:
Ereignis-ID:
Fehlercode:
Vollständige Meldung:
Zeitpunkt:
Auslösende Aktion:
```

Regeln:

- Code vollständig übernehmen,
- hexadezimale und dezimale Schreibweise nicht verwechseln,
- Anbieter und Produktversion notieren,
- nur offizielle Herstellerdokumentation als gesicherte Bedeutung behandeln,
- nicht denselben Code aus einem anderen Produkt übertragen,
- innere oder verschachtelte Fehlermeldungen mit erfassen,
- Ursache erst nach Prüfung festlegen.

> Eine Suchmaschinenfundstelle ohne passenden Hersteller-, Versions- und Komponentenkontext ist kein ausreichender Ursachenbeweis.

---

29. Fehlende Protokolle richtig bewerten

Keine gefundene Meldung bedeutet nicht automatisch, dass kein Fehler auftrat.

Mögliche Gründe:

- falsches Zeitfenster,
- falsche Zeitzone,
- falscher Server,

- falsches Protokoll,
- falscher Prozessname,
- Dienst wurde mit neuer PID gestartet,
- Protokollierung ist deaktiviert,
- Aufbewahrung ist abgelaufen,
- Logrotation hat die Datei verschoben,
- Rate-Limiting unterdrückte Meldungen,
- Anwendung schreibt in eine andere Datei,
- Meldung wurde nur auf dem Backend erzeugt,
- Debugebene war nicht aktiviert,
- Container wurde entfernt,
- Dienst stürzte vor Initialisierung der Protokollierung ab,
- Berechtigung zum Lesen fehlt.

> Das Fehlen einer Meldung ist nur dann aussagekräftig, wenn bekannt ist, dass genau dieses Ereignis an dieser Stelle protokolliert werden müsste.

---

30. Sensible Inhalte in Protokollen erkennen

Protokolle können enthalten:

- Benutzernamen,
- E-Mail-Adressen,
- IP-Adressen,
- interne Servernamen,
- Dateipfade,
- Dokumentnamen,
- Datenbankabfragen,
- URL-Parameter,
- Sitzungscookies,
- Zugriffstoken,
- API-Schlüssel,
- Authorization-Header,
- personenbezogene Daten,
- interne Geschäfts- und Systeminformationen.

Vor Weitergabe prüfen:

```
[ ] Zweck der Weitergabe geklärt
[ ] Empfänger berechtigt
[ ] Zeitraum auf das Notwendige begrenzt
[ ] Nicht relevante Ereignisse entfernt
[ ] Kennwörter und Token entfernt
[ ] Cookies und Authorization-Header entfernt
[ ] Personenbezogene Daten geschützt
[ ] Interne Infrastrukturinformationen bewertet
[ ] Originaldatei beweissicher aufbewahrt
[ ] Bearbeitete Kopie als solche gekennzeichnet
```

> Das Original darf bei einer Beweissicherung nicht unkontrolliert verändert werden. Für Weitergaben wird eine gesonderte bereinigte Kopie erstellt.

---

31. Relevante Protokollstellen dokumentieren

Dokumentationsvorlage:

```
Störung:
Server:
Dienst:
Betriebssystem:
Zeitzone:
Untersuchter Zeitraum:
Protokollquelle:
Ereignisanbieter beziehungsweise Prozess:
Ereignis-ID:
Fehlercode:
Zeitstempel:
Vollständige relevante Meldung:
Unmittelbar vorhergehendes Ereignis:
Unmittelbar folgendes Ereignis:
Betroffene Abhängigkeit:
Interpretation:
```

Beleg für die Interpretation:

Nächster Prüfschritt:

Gute Dokumentation:

> Um 09:15:02 Uhr protokollierte der Anwendungsdienst eine zurückgesetzte Datenbankverbindung. Zwei Sekunden zuvor meldete der Datenbankserver einen Schreibfehler. Die zeitliche Reihenfolge spricht für einen Zusammenhang, der durch Datenträger- und Datenbankprüfung bestätigt werden muss.

Unzureichende Dokumentation:

> In den Logs stand etwas mit Datenbank.

---

32. Typische Fehler bei der Protokollanalyse

Fehler	Folge	Bessere Vorgehensweise
Gesamtes Protokoll ohne Zeitfilter lesen	Relevante Ereignisse gehen in Datenmenge unter	Enges Zeitfenster verwenden
Nur Fehlerstufe anzeigen	Relevante Warnungen oder Infos fehlen	Kontext vor und nach dem Fehler lesen
Nur Hauptserver prüfen	Backendursache bleibt verborgen	Abhängige Systeme einbeziehen
Neueste Meldung als Ursache behandeln	Folgefehler wird verwechselt	Zeitlich erstes relevantes Ereignis suchen
Ereignis-ID ohne Anbieter bewerten	Falsche Bedeutung möglich	Anbieter, Protokoll und Version notieren
Zeitzonen ignorieren	Ereigniskette wird falsch sortiert	Zeitstempel normalisieren
Debuglogging dauerhaft aktivieren	Datenmenge und Datenschutzrisiko steigen	Nur kontrolliert und zeitlich begrenzt
Protokoll während Störung löschen	Beweise gehen verloren	Nur lesen und sichern
Vollständiges Archiv ungeprüft versenden	Sensible Daten werden offengelegt	Bereinigte Kopie erstellen

Fehler	Folge	Bessere Vorgehensweise
Einzelne Meldung ohne Funktionstest bewerten	Hypothese bleibt unbestätigt	Gezielten Test durchführen

33. Checkliste zur Protokollanalyse

[ ] Störung und auslösende Aktion eindeutig beschrieben
[ ] Server und Dienst bestätigt
[ ] Beginn und Ende des Fehlers bestimmt
[ ] Zeitzonen aller beteiligten Systeme geprüft
[ ] Zeitfenster einige Minuten erweitert
[ ] Betriebssystemprotokoll geprüft
[ ] Dienst- beziehungsweise Anwendungsprotokoll geprüft
[ ] Ereignisanbieter oder Prozess gefiltert
[ ] Ereignis-ID und Fehlercode erfasst
[ ] Meldungen unmittelbar vor dem Fehler gelesen
[ ] Meldungen unmittelbar nach dem Fehler gelesen
[ ] Abhängige Systeme geprüft
[ ] Ursache und Folge getrennt
[ ] Prozessneustart und PID-Wechsel berücksichtigt
[ ] Protokollaufbewahrung und Rotation geprüft
[ ] Fehlende Meldungen nicht vorschnell bewertet
[ ] Relevante Protokolle gesichert
[ ] Sensible Inhalte vor Weitergabe entfernt
[ ] Hypothese durch einen technischen Test geprüft
[ ] Ergebnis und nächster Schritt dokumentiert

Bewertung des Ergebnisses

Ergebnis	Nächster Schritt
Dienst konnte nicht gestartet werden	Exitcode, Konto, Pfad und Konfiguration prüfen
Dienst wurde unerwartet beendet	Absturzbericht, Ressourcen und Wiederherstellungsaktion prüfen
Abhängigkeit ist nicht erreichbar	DNS-, Netzwerk-, Port- und Backendprüfung durchführen
Zugriff wurde verweigert	Dienstkonto, Datei- und Systemberechtigungen prüfen

Ergebnis	Nächster Schritt
Speicher- oder Datenträgerfehler vorhanden	Ressourcen und Dateisystem untersuchen
TLS- oder Zertifikatsfehler vorhanden	Zertifikat, Name, Zeit und Vertrauenskette prüfen
Protokoll zeigt Zeitüberschreitung	Zielsystem, Last, Netzwerk und Timeoutursache prüfen
Mehrere Systeme zeigen dieselbe Ereigniskette	Gemeinsame Ursache priorisiert untersuchen
Keine Meldung gefunden	Zeitfenster, Quelle, Aufbewahrung und Logziel prüfen
Protokoll liefert nur einen Folgefehler	Zeitlich frühere Komponente untersuchen

---

Merksatz

> **Ein Protokolleintrag ist ein Beweis für eine aufgezeichnete Beobachtung. Erst Zeitfolge, Systemzusammenhang und ein bestätigender Test machen daraus einen belastbaren Ursachenhinweis.**

Weiterführende Quellen

- Microsoft Learn – Get-WinEvent
- Microsoft Learn – Erstellen effizienter Ereignisabfragen mit FilterHashtable
- Microsoft Learn – wevtutil
- Microsoft Learn – Windows-Ereignisanzeige
- systemd – journalctl
- systemd – journald.conf
- systemd – systemd-journald.service
- Apple – log-Handbuchseite
- Apple – Protokollmeldungen in der Konsole anzeigen
- Apple – Protokollmeldungen und Aktivitäten suchen
