

Dokumente Frank

- [Microsoft Authentifikator als Standard und 2 stellige Nummerneingabe erzwingen](#)
- [Libre nms smnp](#)
- [NET-SMNP installieren](#)
- [internes Windows-SMNP installieren](#)
- [Netzwerktopologie SPBL Docusnap 27.11.2025](#)
- [NTLM HARDENING](#)

Microsoft Authenticator als Standard und 2 stellige Nummerneingabe erzwingen

[Microsoft Authenticator als Standard festlegen.pdf](#)

Was du damit schon richtig gemacht hast

- **Zahlenübereinstimmung für Pushbenachrichtigungen erforderlich: Aktiviert** (grau/erzwungen) → sehr gut, das reduziert „MFA-Fatigue“ deutlich.
- **Anwendungsname anzeigen: Aktiviert** → Nutzer sehen, *welche* App gerade authentifiziert werden soll.
- **Geografischen Standort anzeigen: Aktiviert** → hilft, „unmögliche“ Logins schneller zu erkennen.
- Ziel jeweils **Alle Benutzer** → grundsätzlich ok (Break-Glass später separat behandeln).

Das ist ein guter Basisschutz – aber: Gegen die von dir beschriebenen „SharePoint-Download“-Phishing-Seiten (AiTM/MFA-Phishing) ist das **nicht ausreichend**, weil Angreifer trotzdem in Echtzeit die Anmeldung + MFA „durchreichen“ und ggf. Session-Tokens abgreifen können.

Nächster Schritt: OTP-Fallback bewusst entscheiden

In deinem Screenshot sehe ich „**OTP zulassen**“ (das ist der „Code anzeigen“-Fallback in Authenticator).

- **Wenn du maximal gegen Phishing härten willst:** setze „**OTP zulassen**“ = **Nein** (oder zumindest nicht breit für alle), *weil OTP sehr leicht in Phishing-Seiten abgefragt werden kann.*
- **Wenn du OTP brauchst (Offline/Notfälle):** lass es an, aber dann solltest du parallel „stärkere“ Wege etablieren (siehe unten), damit Nutzer nicht automatisch auf OTP

ausweichen.

Wichtig: Wenn du OTP deaktivierst, stelle sicher, dass du **mindestens** einen sauberen Wiederherstellungs-/Notfallpfad hast (z. B. Temporary Access Pass, zusätzlicher zweiter Faktor, Break-Glass-Konten).

Der wirksamste Hebel gegen eure SharePoint-MFA-Phishing-Welle

1) Conditional Access: Authentifizierungsstärke „Phishing-resistent“ für SharePoint/Office

Setze für **SharePoint Online / Office 365** eine CA-Richtlinie, die nicht nur „MFA“, sondern eine **phishing-resistente Authentifizierung** verlangt (z. B. **Windows Hello for Business** oder **FIDO2 Security Key**).

Damit brichst du genau das Angriffsmuster, bei dem die Phishing-Seite MFA „mit abfragt“.

Pfad: Microsoft Entra Admin Center → **Schutz** → **Bedingter Zugriff** → **Richtlinien**

- **Cloud-Apps:** *Office 365* oder *SharePoint Online*
- **Zugriffssteuerungen (Gewähren):** **Authentifizierungsstärke** = *Phishing-resistent*
- Pilot zuerst nur mit Admins/IT + ausgewählten Power-Usern, dann ausrollen.

2) Conditional Access: Zugriff nur von „verwalteten“ Geräten (optional, sehr effektiv)

Wenn ihr Intune/Managed Devices habt: für SharePoint/Office im Browser zusätzlich

- **„Gerät muss als kompatibel markiert sein“** oder **Hybrid Azure AD-verbunden**
Das reduziert Token-Diebstahl-Szenarien massiv, weil der Angreifer i. d. R. nicht von einem „compliant“ Gerät kommt.

3) Defender for Office 365: Safe Links (falls lizenziert)

Da eure Angriffe über „Downloadangebote“ kommen: **Safe Links** ist der praktische Schutzhebel im Mail-/Link-Kanal (Links werden beim Klicken geprüft/umgeschrieben).

Quick-Check zu deiner aktuellen Konfiguration

Wenn du das noch nicht gemacht hast, prüfe zusätzlich in Entra:

- **Welche MFA-Methoden sind erlaubt?** (SMS/Anruf nach Möglichkeit raus, Authenticator/FIDO2/WHfB bevorzugen)
- **Break-Glass:** 1-2 Notfallkonten, sauber dokumentiert, mit separatem Schutz (keine alltägliche Nutzung), und in CA-Policies korrekt ausgeschlossen.

Wenn du mir sagst, ob ihr **Entra ID P1/P2** und/oder **Intune** habt, kann ich dir eine konkrete Zielkonfiguration skizzieren (Pilotgruppen, genaue CA-Policy-Settings, sinnvolle Ausnahmen), die genau auf euren „SharePoint-Phishing“-Use-Case zielt.

Break-Glass: 1-2 Notfallkonten, sauber dokumentiert, mit separatem Schutz (keine alltägliche Nutzung), und in CA-Policies korrekt ausgeschlossen. Erkläre bitte detailliert!

Break-Glass- bzw. **Notfallkonten** (Microsoft nennt sie „Emergency access accounts“) sind **separat angelegte Global-Admin-Konten**, die du **nur** nutzt, wenn du dich durch eine Fehlkonfiguration (z. B. Conditional Access) oder einen externen Ausfall sonst aus dem Tenant aussperrst. Microsoft empfiehlt dafür **mindestens zwei** solche Konten. [Microsoft Learn](#)

Unten ist ein praxistaugliches, detailliertes Setup, das sowohl **Lockout-Risiken** minimiert als auch **Sicherheitsanforderungen** berücksichtigt.

1) Ziele und Grundregeln

Warum überhaupt Break-Glass?

Typische Notfälle:

- Conditional-Access-Policy blockiert versehentlich alle Admins
- MFA-/Authenticator-Problem (Push-Ausfall, Registrierungsprobleme)
- Föderation/Identity Provider down (bei federierten Tenants) [Microsoft Learn](#)

Grundregeln

- **Kein Daily-Use.** Nutzung nur für Tests und echte Notfälle. [Microsoft Learn](#)
 - **Mindestens 2 Konten** (Redundanz). [Microsoft Learn](#)
 - **Aus allen Conditional-Access-Policies ausschließen**, sonst riskierst du Tenant-Lockout. [Microsoft Learn+1](#)
 - **Nutzung überwachen und alarmieren** (jeder Login ist ein Incident/Change). [Microsoft Learn](#)
-

2) Empfohlenes Kontodesign (2 Konten mit unterschiedlichen Eigenschaften)

Ich empfehle dieses **Dual-Design**:

Konto BG-01 „Maximum Availability“ (reiner Notfallzugang)

- Cloud-only Benutzer (nicht federiert)
- Sehr langes, zufälliges Passwort (z. B. 32-64 Zeichen)
- **Globaler Administrator**
- **Von allen CA-Richtlinien ausgeschlossen** (wirklich *allen*)
- **Keine Produkt-Lizenzen**, kein Postfach, keine Teams-/SharePoint-Nutzung (nur Portal-Zugang)
- Zweck: „Wenn alles brennt, komme ich rein.“

Konto BG-02 „Secure Admin Fallback“ (Notfall + Test + „Portal-MFA“-Realität abfangen)

- Ebenfalls cloud-only
- Globaler Administrator
- Von allen CA-Richtlinien ausgeschlossen
- **Zusätzlich** mit **phishing-resistenter MFA** registriert (z. B. FIDO2/Passkey oder Zertifikat), damit du einen sicheren Admin-Zugang hast, falls du später strengere Baselines/Portal-Anforderungen triffst.
- Zweck: „Sicherer Rückweg und regelmäßiger Funktionstest.“

Microsoft betont: Notfallkonten sollen **nicht** denselben Mechanismus/Abhängigkeiten haben wie normale Admin-Konten und du sollst sie gezielt verwalten. [Microsoft Learn](#)

3) Schritt-für-Schritt Umsetzung

Schritt 1: Konten anlegen

Microsoft Entra Admin Center → Identität → Benutzer → Alle Benutzer → Neuer Benutzer erstellen

- Namen bewusst neutral halten (nicht „breakglass@...“ in großen Umgebungen; in kleineren Umgebungen ist es ok – Hauptsache dokumentiert).
- „Kennwort automatisch generieren“ und **sofort** in einen Passwort-Tresor (siehe unten).

Schritt 2: Rollen zuweisen (Globaler Administrator)

Microsoft Entra Admin Center → Identität → Rollen & Administratoren → Globaler Administrator → Zuweisungen

- BG-01 und BG-02 hinzufügen.

Schritt 3: Von Conditional Access ausschließen (entscheidend)

Für **jede** CA-Richtlinie:

Microsoft Entra Admin Center → **Schutz** → **Bedingter Zugriff** → **Richtlinien** → Richtlinie öffnen

- **Benutzer** → **Ausschließen** → BG-01 und BG-02

Microsoft nennt das explizit als Best Practice: Break-Glass/Emergency Accounts von CA-Policies ausnehmen, um Lockout durch Fehlkonfiguration zu verhindern. [Microsoft Learn](#)

Wichtig: Das gilt auch für **Microsoft-verwaltete Conditional-Access-Richtlinien** – auch dort sollst du Break-Glass-Konten ausschließen. [Microsoft Learn](#)

Und: Microsoft weist darauf hin, dass diese Exclusion-Liste regelmäßig zu auditieren ist. [Microsoft Learn](#)

Schritt 4: Monitoring/Alarmierung festlegen (Pflicht)

Mindestens:

- **Jeder Login** der BG-Konten löst eine Alarmierung an IT/Security aus
- Auswertung über **Anmeldeprotokolle** / Export (z. B. Log Analytics/Sentinel), je nach eurem Setup

Microsoft fordert, diese Konten eng zu überwachen und Nutzung auf echte Notfälle zu beschränken.

[Microsoft Learn](#)

Schritt 5: Sichere Aufbewahrung + Zugriffskontrolle

- Passwort in **Password Vault** (z. B. KeePass/Bitwarden/... oder Firmen-Tresor)
 - Zugriff auf die Secrets nach 4-Augen-Prinzip (z. B. zwei getrennte Personen)
 - Dokumentation: Zweck, wann benutzen, Runbook, wer informieren, Post-Incident Schritte
-

4) Betrieb: Tests, Runbook, Audit

Regelmäßiger Test

- 1× pro Quartal: Login-Test mit BG-02 (und ggf. BG-01 sehr selten), prüfen ob:
 - Zugang zu Entra Admin Center funktioniert
 - Rollen stimmen
 - Alarmierung triggert
- Danach: Ticket/Change dokumentieren.

Incident-Runbook bei echter Nutzung

Wenn BG-Konto benutzt wurde:

1. Grund identifizieren (welche Policy/Änderung hat ausgesperrt?)
2. Sofortige Korrektur (Policy zurückrollen/fixen)
3. BG-Konto wieder „weglegen“
4. Review: Warum kam es dazu? Welche Guardrails fehlen?

Audit der Exclusions

- Monatlich/Quartalsweise: Liste der CA-Policies prüfen, ob BG-01/BG-02 wirklich überall ausgeschlossen sind (neue Policies werden oft vergessen).
-

5) Typische Fehler, die ich in der Praxis sehe

- Break-Glass-Konto existiert, aber **nicht in allen Policies ausgeschlossen**
 - Konto hat eine normale Mailbox und wird „doch mal schnell“ genutzt
 - Passwort liegt in einem persönlichen Notiztool statt in einem kontrollierten Tresor
 - Es gibt **nur eins** - wenn das gesperrt/gelöscht/vergessen ist, ist es vorbei
 - Niemand merkt, dass das Konto genutzt wurde (keine Alarmierung)
-

Wenn du mir sagst, ob ihr **Entra ID P1/P2** habt und ob ihr **Microsoft Sentinel/Log Analytics** nutzt, kann ich dir auch ein konkretes Monitoring-Pattern geben (welche Log-Abfrage/Alarm-Logik

sinnvoll ist) und eine Checkliste, wie du die CA-Exclusions revisionsicher dokumentierst.

Entra ID P1

Alles klar – mit **Entra ID P1** hast du die wichtigste Grundlage: **Bedingter Zugriff** (Conditional Access). Was dir im Vergleich zu P2 fehlt, ist vor allem **PIM (Privileged Identity Management)** und **risikobasierte Policies (Identity Protection)**. Break-Glass funktioniert aber auch mit P1 sehr gut, wenn du es sauber aufsetzt.

Zielbild für Break-Glass mit Entra ID P1

Du brauchst **mindestens 2** Notfallkonten:

1. **BG-01 (Maximum Availability)**

- Soll *immer* funktionieren – auch wenn MFA/Authenticator/Telefonie kaputt ist.
- Typisch: **Passwort-only**, extrem stark, streng verwahrt, **überall von CA ausgeschlossen**.

2. **BG-02 (Secure Fallback / Testkonto)**

- Für regelmäßige Funktionstests und “sicherer” Notfallzugang (z. B. mit FIDO2/Passkey registriert).
- Ebenfalls **von CA ausgeschlossen**, aber du nutzt es im Notfall bevorzugt, wenn alles normal läuft.

Der Gedanke dahinter: **Ein Konto maximiert Verfügbarkeit**, das andere **maximiert Sicherheit** – und du hast Redundanz.

Schritt-für-Schritt Setup

1) Konten anlegen (Cloud-only)

Microsoft Entra Admin Center → **Identität** → **Benutzer** → **Alle Benutzer** → **Neuer Benutzer**

Empfehlungen:

- Cloud-only (nicht an AD gebunden, nicht federiert)
- Keine Lizenz, kein Postfach (wenn möglich)

- Anzeigename eindeutig (z. B. "Emergency Access 01/02"), UPN muss nicht "breakglass" heißen, aber intern klar dokumentiert.

2) Rollen zuweisen (Globaler Administrator)

Microsoft Entra Admin Center → **Identität** → **Rollen & Administratoren** → **Globaler Administrator** → **Zuweisungen** → Benutzer hinzufügen

Mit P1 ist das eine **permanente** Rollenzuweisung (kein PIM). Umso wichtiger: keine Alltagsnutzung + starke Kontrollen.

3) Von allen Conditional-Access-Richtlinien ausschließen (kritisch)

Für **jede** CA-Richtlinie:

Microsoft Entra Admin Center → **Schutz** → **Bedingter Zugriff** → **Richtlinien** → Richtlinie öffnen

- **Benutzer** → **Ausschließen** → BG-01 und BG-02 hinzufügen

Wichtig:

- Auch neue Policies später nicht vergessen (das ist die häufigste Lockout-Ursache).
- Wenn du „Vorlagen“ oder „Microsoft-verwaltete“ Richtlinien nutzt: ebenfalls Exclusions prüfen.

4) Passwörter und Aufbewahrung (operativ entscheidend)

Für BG-01 und BG-02:

- Passwort mit hoher Entropie (z. B. 32–64 Zeichen, zufällig generiert)
- Speicherung in einem **Unternehmens-Tresor** (z. B. KeePass im geschützten Share / Passwortmanager mit 4-Augen-Freigabe)
- Zugriff nur für sehr wenige Personen, ideal: **2 Personen getrennt** (Vier-Augen-Prinzip)
- Dokumentiere: *Wer darf öffnen? Wann? Was ist danach zu tun?*

Kennwortablauf:

Wenn eure Policy regelmäßige Passwortwechsel erzwingt: okay, dann muss das im Runbook drin sein (und der Tresor aktuell gehalten werden). Wenn möglich, für Notfallkonten "vernünftig" lösen: sehr starkes Passwort + Rotation nach Nutzung/Incident.

5) MFA-Strategie für Break-Glass (realistisch und praxistauglich)

Hier gibt es zwei legitime Varianten:

Variante A (empfohlen, weil ausfallsicher):

- **BG-01 ohne MFA-Pflicht** (damit du auch reinkommst, wenn MFA nicht geht)
- **BG-02 mit registriertem phishing-resistentem Faktor** (z. B. FIDO2/Passkey), den du im Notfall bevorzugt nutzt

Hinweis: Wenn du MFA *erzwingen* willst, machst du das normalerweise über CA – aber Break-Glass ist ja ausgeschlossen. Deshalb ist die gängige Praxis: **ein Konto ohne MFA-Pflicht, dafür extrem streng überwacht.**

6) "Security Defaults" prüfen (sonst beißt sich das)

Wenn **Sicherheitsstandards** aktiv sind, erzwingen sie MFA breit und du kannst Break-Glass nicht sauber ausnehmen.

Prüfen:

Microsoft Entra Admin Center → **Identität** → **Übersicht** → **Eigenschaften** → **Sicherheitsstandards verwalten**

- Wenn ihr Conditional Access aktiv nutzt: in der Regel sind Security Defaults **deaktiviert** (sonst kollidiert das häufig).

Monitoring und Alarmierung (Pflicht, gerade weil CA

ausgeschlossen)

Mit P1 hast du standardmäßig die **Anmeldeprotokolle**. Das Minimum ist:

A) Sofortige Regel: Jeder Break-Glass-Login = Incident/Change

- Wenn sich BG-01 oder BG-02 anmeldet, muss automatisch ein Alarm an IT/Security gehen.

B) Umsetzungsmöglichkeiten

1. Manuell (Minimum)

Microsoft Entra Admin Center → **Überwachung** → **Anmeldeprotokolle**

Filter nach BG-01/BG-02 und regelmäßig prüfen.

2. Sauber (empfohlen): Azure Monitor / Log Analytics Alert

- Entra Sign-in Logs in **Log Analytics** (Workspace) exportieren
- Alert-Regel: „Wenn UserPrincipalName == BG-01 oder BG-02 und Status == Erfolg/Fehler → E-Mail/Teams/Webhook“

Beispiel-KQL (für den Alert), damit du eine Vorstellung hast:

```
SigninLogs | where UserPrincipalName in ("bg-01@deinedomain.tld","bg-02@deinedomain.tld") | where ResultType == 0 or ResultType != 0 | project TimeGenerated, UserPrincipalName, AppDisplayName, IPAddress, ClientAppUsed, ResultDescription
```

Runbook: Was passiert, wenn Break-Glass genutzt wurde?

Wenn BG benutzt wurde (egal ob Test oder Notfall), standardisiere:

1. Ticket/Change anlegen (Begründung, Zeitpunkt, wer)
2. Ursache beheben (z. B. CA-Policy zurückrollen/fixen)
3. **Passwort sofort rotieren** und Tresor aktualisieren
4. Sign-in Logs prüfen (IP, Land, App, Zeit)
5. Falls Verdacht auf Kompromittierung: zusätzliche Maßnahmen (Sessions widerrufen, weitere Admin-Konten prüfen)

Checkliste zum Abschluss (P1-tauglich)

- 2 cloud-only Notfallkonten existieren
- Globaler Administrator zugewiesen (permanent, da kein PIM)
- In **jeder** CA-Policy unter **Benutzer** → **Ausschließen** enthalten
- Kein Postfach / keine Lizenz (wenn möglich)
- Passwort extrem stark, im Tresor, 4-Augen-Prozess
- Monitoring/Alerting aktiv (mindestens Sign-in Logs, besser Log Analytics Alert)
- Quartalsweiser Funktionstest (bevorzugt BG-02)

Wenn du willst, beschreib kurz eure aktuelle CA-Struktur (z. B. "MFA für alle", "Block legacy auth", "nur compliant devices"), dann kann ich dir eine konkrete Empfehlung geben, **wo** genau du die Break-Glass-Exclusions am häufigsten vergisst und wie du das auditierbar machst.

Libre nms smnp

Ändert librenms bei smnp scans die ip adressen zuordnung von smnp geräten wenn sie sich ändern?

Das kommt ganz darauf an, **wie** du das Gerät ursprünglich zu LibreNMS hinzugefügt hast.

Die kurze Antwort lautet: **Automatisch passiert das in der Regel nur, wenn du DNS nutzt.**

Wenn du Geräte per IP hinzugefügt hast, wird LibreNMS das Gerät als "Down" markieren, wenn sich die IP ändert, anstatt die Adresse automatisch zu aktualisieren.

Hier ist die Aufschlüsselung, wie LibreNMS auf deinem Proxmox-Cluster damit umgeht:

1. Szenario: Gerät wurde per DNS-Hostname hinzugefügt (Empfohlen)

Wenn du das Gerät z. B. als `server01.deine-domain.local` hinzugefügt hast:

- **Ja, es funktioniert.** LibreNMS löst bei jedem Polling- oder Discovery-Lauf den Hostnamen neu auf.
- Ändert sich die IP-Adresse im DNS, bemerkt LibreNMS dies beim nächsten Lauf, aktualisiert die zugeordnete IP in der Datenbank und das Monitoring läuft nahtlos weiter.

2. Szenario: Gerät wurde per IP-Adresse hinzugefügt

Wenn du das Gerät z. B. als `192.168.1.50` hinzugefügt hast:

- **Nein, es wird nicht automatisch aktualisiert.** LibreNMS pollt stur diese spezifische IP.
- Wechselt das Gerät auf `192.168.1.51`, erhält LibreNMS keine Antwort mehr von der `.50` und meldet das Gerät als **Down**.
- Selbst wenn ein "SNMP Scan" (Auto-Discovery) über das Subnetz läuft und das Gerät unter der neuen IP `.51` findet, wird LibreNMS es meist als **neues Gerät** erkennen (oder wegen

Duplikat-Schutz meckern), aber nicht den alten Eintrag "verschieben".

Besonderheit bei Net-SNMP (Windows)

Da du Net-SNMP unter Windows nutzt, hast du einen Vorteil bei der Identifikation. Net-SNMP liefert in der Regel einen sauberen `sysName` (System Name) zurück.

- LibreNMS nutzt diesen `sysName`, um Duplikate zu erkennen.
- Wenn du **Auto-Discovery** für dein Subnetz aktiviert hast, wird LibreNMS das Gerät unter der neuen IP finden. Es wird dann aber wahrscheinlich warnen, dass ein Gerät mit diesem `sysName` bereits existiert (unter der alten IP), anstatt die alte IP einfach zu überschreiben. Das dient dem Schutz vor Datenkorruption (damit nicht RRD-Graphen von zwei verschiedenen Geräten vermischt werden).

Lösungsmöglichkeiten

Wenn sich eine IP geändert hat und das Gerät in LibreNMS "Down" ist, hast du zwei saubere Wege, das zu korrigieren, ohne die Historie (Graphen) zu verlieren:

A. Über die Web-GUI (Edit Device): Du gehst auf das Gerät -> Einstellungen (Zahnrad) -> **Edit**. Dort kannst du das Feld "Hostname" bzw. "IP" manuell auf die neue IP ändern.

B. Über die Konsole (Proxmox/LXC): Das ist oft schneller und sauberer. Du kannst das eingebaute Rename-Tool nutzen:

Bash

```
cd /opt/librenms
./lnms device:rename <ALTE_IP> <NEUE_IP>
```

(bzw. `./lnms device:rename <ALTE_IP> <NEUER_DNS_NAME>` - was langfristig besser ist).

Zusammenfassung

Verlasse dich bei IP-Änderungen **nicht** auf den automatischen SNMP-Scan. LibreNMS ist hier konservativ, um Datenfehler zu vermeiden. Der sicherste Weg für eine dynamische Umgebung ist immer:

1. **DNS nutzen** (und TTL niedrig halten).
 2. Geräte in LibreNMS per **FQDN** hinzufügen.
-

Möchtest du wissen, wie man ein Skript aufsetzt, das IP-Änderungen über die Net-SNMP `sysName` Identifizierung automatisch in LibreNMS nachpflegt (via API)?

Kann ich erkannte client geräte mit net-snmp automatisch mit fqdn hinzufügen?

Die direkte Antwort ist: **Ja, aber nur indirekt über DNS.**

LibreNMS hat keine eingebaute Funktion "Nutze `sysName` als Hostnamen beim Hinzufügen". Es verlässt sich beim Auto-Discovery (SNMP Scan) fast vollständig auf die Namensauflösung (DNS).

Hier sind die drei Wege, wie du das Ziel erreichst, dass deine Net-SNMP Clients mit FQDN statt IP erscheinen:

1. Der "Königsweg": Reverse DNS (PTR Records)

Das ist der Weg, den LibreNMS standardmäßig erwartet.

- **Wie es funktioniert:** Wenn der Discovery-Scanner eine IP (z.B. `192.168.1.50`) findet, fragt er deinen DNS-Server: "Wer ist `.50`?"
- **Voraussetzung:** Du musst im DNS-Server (in deinem Fall vermutlich auf dem Windows Domain Controller oder dem Proxmox-Umfeld) einen **Reverse-Lookup-Zone (PTR)** Eintrag für die Clients haben.
- **Ergebnis:** Wenn der DNS-Server mit `client01.deine-domain.local` antwortet, fügt LibreNMS das Gerät automatisch mit diesem FQDN hinzu. Antwortet DNS nicht, wird die IP genommen.

2. Die "Visuelle" Lösung (Empfohlen, wenn kein Reverse DNS möglich)

Wenn du keinen sauberen Reverse-DNS für alle Clients pflegen willst, kannst du LibreNMS anweisen, die IP in der Datenbank zu behalten, aber in der Oberfläche **immer** den `sysName` (den Net-SNMP liefert) anzuzeigen.

Füge dazu folgende Zeile in deine `/opt/librenms/config.php` ein:

PHP

```
$config['force_ip_to_sysname'] = true;
```

- **Vorteil:** Die Geräte werden technisch per IP überwacht (keine DNS-Probleme), aber in Listen, Graphen und Warnungen siehst du den echten Namen (z.B. `DESKTOP-H3K9L`).
- **Nachteil:** In der Datenbank ist der "Hostname" weiterhin die IP.

3. Die "Harte" Methode (Automatisiertes Umbenennen)

Da du Net-SNMP unter Windows nutzt, liefert das Gerät zuverlässig seinen Namen. Du kannst ein kleines Skript (Cronjob) laufen lassen, das nachträglich IPs in echte Hostnamen umwandelt, indem es die LibreNMS API oder CLI nutzt.

Beispiel-Logik für ein Skript:

1. Hole Liste aller Geräte, die als IP hinzugefügt sind (Regex auf IP-Muster).
2. Frage per SNMP den `sysName` ab.
3. Führe `./lnms device:rename <IP> <sysName>` aus.

Meine Empfehlung für deine Umgebung

Da du Net-SNMP auf Windows nutzt und das Monitoring auf Proxmox läuft:

1. Aktiviere `$config['force_ip_to_sysname'] = true;`. Das ist sofort wirksam und löst das "Hässliche-IP-Problem" in der Ansicht.
2. Wenn du echte FQDNs in der Datenbank brauchst (z.B. für externe Reports), Sorge dafür, dass dein DHCP-Server die Hostnamen der Windows-Clients automatisch im DNS registriert (Dynamic DNS Updates). Dann macht LibreNMS den Rest beim nächsten Scan von alleine.

erstelle mir ein script das dies 1 mal um 01.00Uhr durchführt für librenms im docker container

Hier ist ein robustes **Bash-Skript**, das auf deinem Host-System (z. B. der VM oder dem LXC-Container in deinem Proxmox-Cluster, in dem Docker läuft) ausgeführt wird.

Es nutzt die LibreNMS-internen Tools (`lnms tinker` und `lnms device:rename`) im Docker-Container, um sicherzustellen, dass die Datenbank und die RRD-Dateien (Graphen) korrekt migriert werden.

Das Skript: `auto_rename_ips.sh`

Erstelle diese Datei auf dem Host, auf dem der Docker-Container läuft (z.B. unter `/root/scripts/auto_rename_ips.sh`).

Bash

```
#!/bin/bash

# --- KONFIGURATION ---
# Name des LibreNMS Docker Containers (prüfen mit 'docker ps')
CONTAINER_NAME="librenms"

# Domain-Suffix (Optional)
# Falls Net-SNMP nur "server1" liefert, du aber "server1.local" willst.
# Leer lassen "", wenn der sysName von Net-SNMP direkt übernommen werden soll.
DOMAIN_SUFFIX=""
# Beispiel: DOMAIN_SUFFIX=".deine-domain.local"

# --- LOGIK ---

echo "[$(date)] Starte IP-zu-sysName Umbenennung..."

# 1. Liste alle Geräte, die als IP gespeichert sind, aber einen sysName haben
# Wir nutzen 'Inms tinker' (PHP Shell), um direkt die DB zu fragen. Das ist sicherer als SQL von außen.
# Regex sucht nach IP-Muster (x.x.x.x)
DEVICES_TO_RENAME=$(docker exec -i $CONTAINER_NAME Inms tinker --execute
"\App\Models\Device::whereRaw(\"hostname REGEXP '^([0-9]{1,3})\\.([0-9]{1,3})\\.([0-9]{1,3})\\.([0-9]{1,3})\$'\")-
>whereNotNull('sysName')->where('sysName', '!=', '')->get()->each(function(\$d) { echo \$d->hostname . '|' .
\$d->sysName . PHP_EOL; });" | grep "|")

# Prüfen, ob Geräte gefunden wurden
if [ -z "$DEVICES_TO_RENAME" ]; then
    echo "Keine Geräte gefunden, die umbenannt werden müssen."
    exit 0
fi

# 2. Loop durch die Ergebnisse und benenne um
while IFS='|' read -r OLD_IP SYS_NAME; do
    # Bereinigen von möglichen Leerzeichen
    SYS_NAME=$(echo "$SYS_NAME" | tr -d '[:space:]')

    # Zieldomain bauen
    NEW_HOSTNAME="${SYS_NAME}${DOMAIN_SUFFIX}"
```

```
# Sicherheitscheck: Nicht umbenennen, wenn Name == IP (passiert manchmal bei Fehlkonfiguration)
if [[ "$NEW_HOSTNAME" == "$OLD_IP" ]]; then
    echo "Überprüfe $OLD_IP: sysName ist identisch mit IP."
    continue
fi

echo "Benenne um: $OLD_IP -> $NEW_HOSTNAME"

# Führe den offiziellen Rename-Befehl im Container aus
docker exec $CONTAINER_NAME lnms device:rename "$OLD_IP" "$NEW_HOSTNAME"

done <<< "$DEVICES_TO_RENAME"

echo "[$(date)] Fertig."
```

Installation & Einrichtung

1. Skript speichern und ausführbar machen

Führe auf deinem Docker-Host folgende Befehle aus:

Bash

```
mkdir -p /root/scripts
nano /root/scripts/auto_rename_ips.sh
# -> Inhalt von oben einfügen und speichern (Strg+O, Enter, Strg+X)

chmod +x /root/scripts/auto_rename_ips.sh
```

2. Testlauf (Trockenübung)

Führe das Skript einmal manuell aus, um zu sehen, ob es funktioniert:

Bash

```
/root/scripts/auto_rename_ips.sh
```

Wenn deine Net-SNMP Clients sauber konfiguriert sind, solltest du jetzt sehen, wie die IPs in Hostnamen umgewandelt werden.

3. Cronjob einrichten (01:00 Uhr nachts)

Öffne die Crontab des Root-Users auf dem Host:

Bash

```
crontab -e
```

Füge folgende Zeile am Ende ein:

Code-Snippet

```
0 1 * * * /root/scripts/auto_rename_ips.sh >> /var/log/librenms_rename.log 2>&1
```

Wichtige Hinweise für deine Umgebung

- Net-SNMP sysName:** Da du Windows mit Net-SNMP nutzt, wird der Name meist als einfacher Hostname (z.B. `TERMINALSERVER01`) zurückgegeben, oft ohne `.domain.local`.
 - Wenn du volle FQDNs willst, trage deine Domain oben im Skript bei `DOMAIN_SUFFIX=".local"` (oder was dein Cluster nutzt) ein.
- Docker Container Name:** Das Skript geht davon aus, dass dein Container `librenms` heißt. Wenn du `docker-compose` nutzt, heißt er oft `librenms_librenms_1` oder ähnlich. Prüfe das kurz mit `docker ps` und passe die Variable `CONTAINER_NAME` im Skript an.
- RRD Daten:** Da wir `Inms device:rename` nutzen, bleiben alle deine historischen Graphen erhalten. Die RRD-Dateien werden auf Dateiebene vom System automatisch umbenannt.

Möchtest du, dass ich das Skript anpasse, um automatisch zu erkennen, ob es sich um einen deiner **Terrascaler Datacore Cluster** Server handelt (z.B. anhand der `sysDescr`), um diese speziell zu behandeln?

NET-SMNP installieren

[install_snmp.ps1](#)

[snmpd.conf](#)

internes Windows-SMNP installieren

[configure-windows-smtp-local.ps1](#)

Netzwerktopologie SPBL

DocuSnap 27.11.2025

[Plan_Topologie \(1\).vsdx](#)

[Spiegelblank Netzwerk Topologie V0.32 bck-gridzize.drawio](#)

NTLM HARDENING