7.9 Ping funktioniert, Anwendung nicht

Ein Server antwortet auf einen Ping, die eigentliche Anwendung ist jedoch nicht erreichbar oder funktioniert nicht vollständig.

Dieses Fehlerbild entsteht häufig, weil ein erfolgreicher Ping fälschlicherweise als Nachweis für die Funktionsfähigkeit des gesamten Dienstes interpretiert wird. Ping prüft jedoch nur einen begrenzten Teil der Kommunikationskette.

7.9.1 Typisches Fehlerbild

Mögliche Meldungen und Beobachtungen:

- Der Server antwortet auf `ping`, aber die Webseite öffnet sich nicht.
- Die IP-Adresse ist erreichbar, aber die Anwendung meldet einen Timeout.
- Der TCP-Port ist erreichbar, aber die Anmeldung schlägt fehl.
- Der Dienst funktioniert lokal auf dem Server, jedoch nicht von einem Client.
- Die Anwendung funktioniert ohne VPN, aber nicht über den VPN-Tunnel.
- Einige Benutzer können zugreifen, andere nicht.
- IPv4 funktioniert, IPv6 jedoch nicht.
- Der Reverse Proxy antwortet, erreicht aber das Backend nicht.
- Die Anwendung zeigt `502 Bad Gateway` oder `503 Service Unavailable`.
- Die Startseite funktioniert, ein bestimmter Geschäftsprozess jedoch nicht.
- Ein Healthcheck ist erfolgreich, während Benutzerfunktionen fehlschlagen.

Die Diagnose muss deshalb über ICMP hinausgehen und den tatsächlichen Anwendungsweg prüfen.

7.9.2 Diagnoseziel

Ziel ist es, eindeutig festzustellen:

- welches Ziel die Anwendung tatsächlich verwendet;
- welche IP-Adresse verwendet wird;
- ob IPv4 oder IPv6 verwendet wird;
- welches Transportprotokoll erforderlich ist;
- welcher Port angesprochen wird;
- ob der Zielport erreichbar ist;
- ob auf dem Server ein passender Prozess lauscht;
- ob der Dienst an der richtigen Adresse gebunden ist;
- ob TLS und Zertifikatsprüfung funktionieren;
- ob ein Proxy, Reverse Proxy oder Load Balancer beteiligt ist;
- ob Authentifizierung und Autorisierung funktionieren;
- ob alle Backendabhängigkeiten verfügbar sind;

- an welcher Stelle die vollständige Kommunikationskette unterbrochen wird.

Ein möglicher Fehlerbereich darf erst dann als Ursache gelten, wenn er durch ein reproduzierbares Prüfergebnis, ein Protokoll oder einen Paketmitschnitt bestätigt wurde.

7.9.3 Kommunikationskette der Anwendung

Eine typische Anwendungsverbindung kann mehrere voneinander unabhängige Prüfebenen enthalten:

1. Anwendungskonfiguration
2. Namensauflösung
3. Auswahl von IPv4 oder IPv6
4. lokales Routing
5. VPN oder Proxy
6. Clientfirewall
7. Netzwerkfirewall oder ACL
8. NAT oder Portweiterleitung
9. Load Balancer oder Reverse Proxy
10. TCP- oder UDP-Kommunikation
11. TLS-Handshake
12. Anwendungsprotokoll
13. Authentifizierung
14. Autorisierung
15. Backenddienste
16. vollständiger Benutzerablauf

Ein Fehler auf einer späteren Ebene kann auftreten, obwohl alle vorherigen Ebenen funktionieren.

Beispiele:

- Ping funktioniert, aber TCP 443 wird blockiert.
- TCP 443 funktioniert, aber der TLS-Handshake schlägt fehl.
- TLS funktioniert, aber der HTTP-Pfad ist falsch.
- HTTP funktioniert, aber die Anmeldung wird verweigert.
- Die Anmeldung funktioniert, aber die Datenbank ist nicht erreichbar.
- Der Healthcheck funktioniert, aber der eigentliche Geschäftsprozess schlägt fehl.

7.9.4 Was Ping tatsächlich prüft

Ping verwendet normalerweise ICMP-Echo-Anfragen und ICMP-Echo-Antworten.

Ein erfolgreicher Ping bestätigt für den konkreten Testzeitpunkt grundsätzlich:

- der verwendete Zielname konnte für diesen Aufruf aufgelöst werden, sofern ein Name angegeben wurde;
- eine Zieladresse wurde ausgewählt;
- der Client konnte ein ICMP-Echo-Paket absenden;
- das Paket erreichte ein antwortendes System;
- eine ICMP-Echo-Antwort erreichte den Client;
- der ICMP-Hin- und Rückweg funktionierte für diese Paketgröße und diesen Zeitpunkt.

Ping kann zusätzlich Hinweise liefern auf:

- Antwortzeit;
- Paketverlust;
- wechselnde Antwortzeiten;
- ausgewählte IPv4- oder IPv6-Adresse;
- grobe Erreichbarkeit eines Systems.

Das antwortende System muss jedoch nicht zwingend der erwartete Anwendungsserver sein. Bei virtuellen IP-Adressen, Load Balancern, Firewalls oder falsch aufgelösten Namen kann ein anderes System antworten.

7.9.5 Was Ping nicht beweist

Ein erfolgreicher Ping beweist nicht:

- dass ein bestimmter TCP-Port geöffnet ist;
- dass ein UDP-Dienst funktioniert;
- dass der erwartete Prozess läuft;
- dass der Dienst an der richtigen Adresse lauscht;
- dass TLS funktioniert;
- dass das Zertifikat gültig ist;
- dass der richtige virtuelle Host ausgewählt wird;
- dass ein Reverse Proxy das Backend erreicht;
- dass ein Load Balancer ein gesundes Backend besitzt;
- dass ein Proxy die Verbindung zulässt;
- dass eine Anmeldung möglich ist;
- dass der Benutzer ausreichend berechtigt ist;
- dass Datenbank, API oder Verzeichnisdienst verfügbar sind;
- dass größere Pakete oder Datenübertragungen funktionieren;
- dass die eigentliche Anwendung dieselbe IP-Adresse wie Ping verwendet;
- dass der vollständige Benutzerablauf funktioniert.

Ebenso beweist ein fehlgeschlagener Ping nicht automatisch, dass der Server oder die Anwendung ausgefallen ist. ICMP kann absichtlich blockiert, begrenzt oder niedriger priorisiert werden, während der eigentliche Anwendungsdienst weiterhin erreichbar ist.

7.9.6 Fehlerumfang bestimmen

Vor technischen Änderungen muss der Umfang des Fehlers festgestellt werden.

Zu prüfen sind:

- Ist nur ein Benutzer betroffen?
- Ist nur ein Client betroffen?
- Ist nur ein Betriebssystem betroffen?
- Ist nur ein Standort oder VLAN betroffen?
- Tritt der Fehler nur über VPN auf?
- Tritt der Fehler nur bei WLAN oder nur bei LAN auf?
- Sind alle Benutzer betroffen?
- Funktioniert die Anwendung intern, aber nicht extern?
- Funktioniert sie über IPv4, aber nicht über IPv6?
- Betrifft der Fehler nur einen bestimmten Funktionsbereich?
- Tritt der Fehler dauerhaft oder nur zeitweise auf?
- Seit welchem Zeitpunkt besteht der Fehler?
- Welche Änderung erfolgte unmittelbar davor?

Geeignete Vergleichstests:

Vergleich	Mögliche Eingrenzung
gleicher Benutzer an anderem Client	clientbezogener Fehler
anderer Benutzer am gleichen Client	benutzerbezogener Fehler
gleicher Client in anderem Netz	Netzwerkpfad oder standortbezogene Regel
Zugriff mit und ohne VPN	VPN-Route, DNS, Proxy, Firewall oder MTU
IPv4 und IPv6 getrennt	adressfamilienbezogener Fehler
direkter Anwendungstest und Zugriff über Proxy	Proxy- oder Reverse-Proxy-Fehler
lokaler Test auf dem Server und entfernter Test	Dienstfehler oder externer Netzwerkpfad
einfacher Endpunkt und vollständiger Benutzerablauf	Backend- oder Anwendungsfehler

Kreuztests dürfen nur kontrolliert durchgeführt werden. Produktive Zugangsdaten oder vertrauliche Daten dürfen nicht ungeschützt in Diagnoseausgaben übernommen werden.

7.9.7 Ausgangszustand sichern

Vor einem Neustart oder einer Konfigurationsänderung sollten mindestens folgende Informationen gesichert werden:

- genaue Fehlermeldung;
- Screenshot oder vollständiger Fehlertext;

- Zeitpunkt einschließlich Zeitzone;
- betroffener Benutzer;
- betroffener Client;
- verwendetes Netzwerk;
- verwendeter Servername;
- verwendete Zieladresse;
- Port und Protokoll;
- URL oder Ressourcenpfad;
- Proxy- und VPN-Zustand;
- relevante Clientprotokolle;
- relevante Serverprotokolle;
- Reverse-Proxy- oder Load-Balancer-Protokolle;
- Dienststatus;
- Listener und Bindungsadresse;
- letzte Änderungen;
- Vergleich mit einem funktionierenden Client.

Ein vorschneller Neustart kann flüchtige Hinweise beseitigen, beispielsweise:

- bestehende Verbindungen;
- Fehlerzustände;
- temporäre Protokolle;
- Speicherauslastung;
- Portbelegung;
- Prozesszustand;
- Warteschlangen;
- reproduzierbare Zeitüberschreitungen.

Änderungen dürfen erst nach Sicherung des Ausgangszustands erfolgen.

7.9.8 Tatsächliches Ziel der Anwendung bestimmen

Vor jedem Port- oder Protokolltest muss geklärt werden, welches Ziel die Anwendung wirklich anspricht.

Zu erfassen sind:

```
Anwendung:
Benutzerfunktion:
Servername:
Vollständiger DNS-Name:
Verwendete IP-Adresse:
IPv4 oder IPv6:
Transportprotokoll:
```

```
Port:
Anwendungsprotokoll:
URL oder Ressourcenpfad:
Proxy:
VPN:
Load Balancer oder Reverse Proxy:
Fehlerzeitpunkt:
```

Die Anwendungskonfiguration kann vom vermuteten Standard abweichen.

Mögliche Einflussquellen:

- explizit eingetragener Servername;
- vollständige URL;
- abweichender Port;
- Hosts-Datei;
- DNS-Suffix;
- Suchdomäne;
- Service-Discovery;
- Umgebungsvariable;
- Proxykonfiguration;
- PAC-Datei;
- Reverse Proxy;
- Load Balancer;
- lokale Anwendungskonfiguration;
- gespeicherte Sitzung oder zwischengespeicherte Adresse;
- IPv4- oder IPv6-Verwendung;
- mandanten-, standort- oder benutzerabhängige Konfiguration;
- Umleitungen innerhalb der Anwendung.

Besonders wichtig:

- Ein Ping auf den Servernamen kann eine andere Adresse verwenden als die Anwendung.
- Ein Proxy kann die eigentliche Verbindung stellvertretend aufbauen.
- Ein Load Balancer kann mehrere Backendserver verwenden.
- Ein Browser kann einen anderen Resolver oder einen eigenen Proxyweg verwenden.
- Eine Anwendung kann eine alte Adresse zwischengespeichert haben.
- Ein direkter Aufruf per IP-Adresse verändert bei HTTPS möglicherweise SNI, Hostname und Zertifikatsprüfung.

Die im Fehlerzeitpunkt verwendete Zieladresse sollte aus der Anwendung, einem Verbindungsprotokoll oder einem Paketmitschnitt bestätigt werden.

7.9.9 Namensauflösung prüfen

Windows:

```
Resolve-DnsName -Name <Servername> -Type A
Resolve-DnsName -Name <Servername> -Type AAAA
```

Linux:

```
getent ahosts <Servername>
dig <Servername> A
dig <Servername> AAAA
```

macOS:

```
dscacheutil -q host -a name <Servername>
dig <Servername> A
dig <Servername> AAAA
```

Zu vergleichen sind:

- die von der Anwendung verwendete Adresse;
- die von `ping` angezeigte Adresse;
- die von `curl -v` verwendete Adresse;
- A- und AAAA-Antworten;
- die Antworten verschiedener DNS-Server;
- interne und öffentliche DNS-Antworten;
- das Verhalten mit und ohne VPN.

Ein erfolgreicher Ping auf einen Namen beweist nur, dass für diesen Ping-Aufruf eine Namensauflösung möglich war. Die Antwort kann aus DNS, einer Hosts-Datei oder einem anderen lokalen Namensauflösungsverfahren stammen.

DNS-Caches sollten nicht als erste Maßnahme gelöscht werden. Zuerst muss dokumentiert werden, welche Adresse aktuell verwendet wird und warum sie falsch sein könnte.

7.9.10 TCP-Port gezielt prüfen

Windows:

```
Test-NetConnection `
  -ComputerName <Servername> `
```

```
  -Port <Port> `
  -InformationLevel Detailed
```

Wichtige Felder:

```
ComputerName
RemoteAddress
RemotePort
InterfaceAlias
SourceAddress
TcpTestSucceeded
```

Linux und macOS:

```
nc -vz <Servername> <Port>
```

Ein erfolgreicher TCP-Test beweist:

- der verwendete Name wurde aufgelöst;
- der Client konnte die verwendete Adresse erreichen;
- der TCP-Verbindungsaufbau zum angegebenen Port wurde abgeschlossen.

Er beweist nicht:

- dass auf dem Port der erwartete Dienst läuft;
- dass TLS funktioniert;
- dass die Anwendung eine gültige Antwort liefert;
- dass eine Anmeldung möglich ist;
- dass der vollständige Geschäftsprozess funktioniert.

7.9.11 TCP-Fehler richtig interpretieren

Ergebnis	Bedeutung	Nächster Nachweis
`TcpTestSucceeded : True`	TCP-Verbindungsaufbau war möglich	Anwendungsprotokoll und TLS prüfen
`Connection refused`	Ziel oder zwischengeschaltetes System hat die Verbindung aktiv abgelehnt	Listener, Port und Serverprotokoll prüfen
Verbindungs-Timeout	Keine rechtzeitige TCP-Antwort	Firewall, Rückweg, Routing, Überlastung oder falsche Adresse untersuchen

Ergebnis	Bedeutung	Nächster Nachweis
Name nicht gefunden	Zielname konnte nicht aufgelöst werden	Namensauflösung prüfen
Verbindung wird sofort getrennt	Dienst lehnt Sitzung ab oder falsches Protokoll wird verwendet	Dienst- und Anwendungsprotokoll prüfen
Verbindung beginnt, bleibt dann hängen	Anwendung, TLS, Backend, Paketverlust oder Pfad-MTU möglich	Protokolle und Paketverlauf untersuchen

Ein Timeout beweist nicht automatisch eine Firewallblockierung. Auch ein falscher Rückweg, ein überlasteter Dienst, ein fehlerhafter Load Balancer oder eine falsche IP-Adresse können zu einem Timeout führen.

7.9.12 Anwendungsprotokoll statt nur Port prüfen

Der Test muss zum tatsächlichen Dienst passen.

Dienst	Geeigneter Funktionstest
HTTP oder HTTPS	`curl -v` mit vollständiger URL
SSH	`ssh -vvv <Benutzer>@<Servername>`
DNS	`Resolve-DnsName` oder `dig` gegen den vorgesehenen DNS-Server
SMB	Zugriff auf die konkrete Freigabe testen
RDP	kontrollierter Verbindungsversuch mit dem RDP-Client
Datenbank	nativen Datenbankclient mit einer sicheren Leseabfrage verwenden
API	vorgesehenen Endpunkt, Methode, Header und Authentifizierung prüfen
herstellerspezifischer Dienst	Diagnoseclient oder dokumentierten Protokolltest des Herstellers verwenden

Ein erfolgreicher Test auf TCP 3389 beweist beispielsweise nur, dass der Port erreichbar ist. Er beweist noch keine funktionierende RDP-Anmeldung oder Sitzung.

`curl` ist für HTTP-, HTTPS- und weitere von curl unterstützte Protokolle geeignet. Ein beliebiger TCP-Dienst darf nicht automatisch mit einer HTTP-Anfrage getestet werden.

7.9.13 HTTP und HTTPS prüfen

```
curl -v --connect-timeout 5 \
  "https://<Servername>:<Port>/<Pfad>"
```

Die Ausgabe wird schrittweise ausgewertet:

1. Welche IP-Adresse wird verwendet?
2. Wird eine TCP-Verbindung aufgebaut?
3. Beginnt der TLS-Handshake?
4. Welches Zertifikat wird präsentiert?
5. Wird die Zertifikatsprüfung erfolgreich abgeschlossen?
6. Welche HTTP-Anfrage wird gesendet?
7. Welcher HTTP-Statuscode wird empfangen?
8. Erfolgt eine Umleitung?
9. Antwortet ein Reverse Proxy oder der erwartete Anwendungsserver?
10. Wird eine Anwendungsfehlermeldung zurückgegeben?

Wichtige HTTP-Ergebnisse:

Status	Einordnung
`2xx`	Anfrage wurde auf HTTP-Ebene erfolgreich verarbeitet
`3xx`	Umleitung; Ziel im `Location`-Header prüfen
`401`	Anwendung erreicht, Authentifizierung erforderlich oder fehlgeschlagen
`403`	Anwendung erreicht, Zugriff wird verweigert
`404`	Dienst antwortet, angeforderter Pfad oder virtuelle Zuordnung fehlt
`407`	Proxy verlangt eine Authentifizierung
`5xx`	Server, Gateway oder Backend meldet einen Fehler

Auch eine gültige HTTP-Fehlerantwort beweist, dass bereits mehrere Prüfebenen erfolgreich durchlaufen wurden. Sie beweist jedoch nicht, dass die benötigte Benutzerfunktion arbeitet.

7.9.14 Bestimmte IP-Adresse mit richtigem Hostnamen testen

Bei HTTPS sollte nicht einfach die IP-Adresse in die URL eingesetzt werden. Dadurch können sich SNI, Host-Header und Zertifikatsprüfung verändern.

Mit curl kann ein Servername kontrolliert einer bestimmten Adresse zugeordnet werden:

```
curl -v \
  --resolve <Servername>:<Port>:<IP-Adresse> \
  "https://<Servername>:<Port>/<Pfad>"
```

Damit bleiben der Servername in der URL, der HTTP-Host und die TLS-SNI-Angabe erhalten, während die Verbindung gezielt zur angegebenen IP-Adresse aufgebaut wird.

Dieser Test ist hilfreich bei:

- mehreren A- oder AAAA-Adressen;
- Load Balancern;
- geplanten DNS-Änderungen;
- einem einzelnen verdächtigen Backend;
- dem Vergleich alter und neuer Zieladressen.

Das Ergebnis gilt nur für die getestete Kombination aus Name, Adresse, Port und Pfad.

7.9.15 TLS kurz prüfen

Für HTTPS oder andere TLS-Dienste:

```
openssl s_client \
  -connect <Servername>:<Port> \
  -servername <Servername> \
  -verify_hostname <Servername> \
  -verify_return_error
```

Zu prüfen sind:

- wird überhaupt eine TCP-Verbindung aufgebaut?
- beginnt der TLS-Handshake?
- wird der richtige Servername per SNI verwendet?
- stimmt der Zertifikatsname mit dem Servernamen überein?
- ist die Zertifikatskette vollständig?
- vertraut der verwendete Client der ausstellenden Zertifizierungsstelle?
- sind Zertifikat und Zwischenzertifikate gültig?
- wird eine gemeinsame TLS-Version beziehungsweise Cipher Suite gefunden?
- verlangt der Server ein Clientzertifikat?

Die Vertrauensstellung von OpenSSL kann sich vom Zertifikatsspeicher der eigentlichen Anwendung unterscheiden. Ein erfolgreicher OpenSSL-Test beweist deshalb nicht automatisch, dass jeder Client dem Zertifikat vertraut.

Zertifikatsprüfungen dürfen nicht dauerhaft mit Optionen wie `-k` oder `--insecure` umgangen werden. Die ausführliche TLS-Diagnose erfolgt auf Seite 7.14.

7.9.16 Proxy und anwendungsspezifischen Verbindungsweg prüfen

Eine Anwendung kann einen Proxy verwenden, obwohl ein direkter Porttest erfolgreich ist. Umgekehrt kann ein direkter Test scheitern, während der Zugriff ausschließlich über einen Proxy

vorgesehen ist.

Windows – WinHTTP-Konfiguration:

```
netsh winhttp show proxy
```

Umgebungsvariablen in PowerShell:

```
Get-ChildItem Env: |
  Where-Object Name -Match '^(HTTP|HTTPS|NO)_PROXY$'
```

Linux:

```
env | grep -iE '^(http|https|no)_proxy='
```

macOS:

```
scutil --proxy


env | grep -iE '^(http|https|no)_proxy='
```

Zusätzlich zu prüfen:

- eigene Proxyeinstellung der Anwendung;
- Browser- oder Benutzerproxy;
- PAC-Datei;
- Proxy-Bypassliste;
- Proxy-Authentifizierung;
- Abweichung zwischen WinHTTP und benutzerbezogenen Einstellungen;
- unterschiedliche Regeln für interne und externe Namen;
- VPN-abhängige Proxykonfiguration.

`netsh winhttp show proxy` zeigt nur die WinHTTP-Konfiguration. Daraus darf nicht automatisch auf die Konfiguration jedes Browsers oder jeder Anwendung geschlossen werden.

7.9.17 Listener auf dem Server prüfen

Wenn der TCP-Port von außen nicht erreichbar ist, muss geprüft werden, ob der Dienst auf dem Server tatsächlich lauscht.

Windows:

```
Get-NetTCPConnection `
  -LocalPort <Port> `
  -State Listen |
  Select-Object LocalAddress, LocalPort, State, OwningProcess
```

Zugehörigen Prozess prüfen:

```
Get-Process -Id <PID>
```

Linux:

```
sudo ss -lntp
```

macOS:

```
sudo lsof -nP \
  -iTCP:<Port> \
  -sTCP:LISTEN
```

Zu prüfen sind:

- richtiger Port;
- erwarteter Prozess;
- lokale Bindungsadresse;
- IPv4- oder IPv6-Bindung;
- Dienststatus;
- mehrere konkurrierende Prozesse;
- unerwartete Portänderung.

Typische Bindungsbefunde:

Bindungsadresse	Bedeutung
`127.0.0.1`	nur lokale IPv4-Verbindungen
`::1`	nur lokale IPv6-Verbindungen
konkrete Serveradresse	nur über diese Adresse beziehungsweise Schnittstelle
`0.0.0.0`	alle lokalen IPv4-Adressen
`[::]`	alle lokalen IPv6-Adressen; zusätzliches IPv4-Verhalten ist systemabhängig

Ein Dienst, der ausschließlich auf `127.0.0.1` lauscht, kann lokal funktionieren und trotzdem von entfernten Clients nicht erreichbar sein.

7.9.18 Lokalen und entfernten Zugriff vergleichen

Auf dem Server kann zunächst ein lokaler Anwendungstest durchgeführt werden:

```
curl -v \
  "http://127.0.0.1:<Port>/<Pfad>"
```

Bei einem namensabhängigen HTTPS-Dienst:

```
curl -v \
  --resolve <Servername>:<Port>:127.0.0.1 \
  "https://<Servername>:<Port>/<Pfad>"
```

Der lokale Test muss an die tatsächliche Listeneradresse angepasst werden. Ein Dienst, der nur an eine bestimmte Serveradresse gebunden ist, muss über diese Adresse geprüft werden.

Lokaler Test	Entfernter Test	Wahrscheinlicher Bereich
schlägt fehl	schlägt fehl	Dienst, Listener, Konfiguration oder Backend
funktioniert	TCP-Port schlägt fehl	Listenerbindung, Hostfirewall oder Netzwerkpfad
funktioniert	TCP-Port funktioniert, Protokoll schlägt fehl	TLS, Reverse Proxy, virtueller Host oder Anwendung
funktioniert	`curl` vom Client funktioniert	Problem im eigentlichen Client oder Benutzerkontext
funktioniert nur über Loopback	entfernt nicht erreichbar	falsche Listenerbindung wahrscheinlich
funktioniert über eine Serveradresse	über andere Adresse nicht	Schnittstelle, Routing, DNS oder Firewallregel

Ein lokaler Erfolg beweist nicht, dass die vollständige externe Zugriffskette funktioniert.

7.9.19 Firewall, ACL, NAT und Load Balancer prüfen

ICMP und Anwendungsverkehr können durch unterschiedliche Regeln behandelt werden.

Zu prüfen sind:

- lokale Firewall auf dem Client;
- Hostfirewall auf dem Server;
- Netzwerkfirewall zwischen den Systemen;
- VLAN- oder Segment-ACL;
- VPN-Regel;
- NAT- oder Portweiterleitung;
- Security Group oder Cloud-Firewall;
- Load-Balancer-Listener;
- Load-Balancer-Healthcheck;
- Reverse-Proxy-Route;
- Quellnetz- oder benutzerabhängige Regel;
- Rückweg vom Server zum Client.

Mögliche Konstellationen:

- ICMP ist erlaubt, TCP 443 wird verworfen.
- TCP 443 erreicht den Load Balancer, aber kein Backend ist gesund.
- Der Reverse Proxy besitzt keinen passenden virtuellen Host.
- Eine NAT-Regel leitet auf den falschen internen Port weiter.
- Eine Firewall erlaubt nur bestimmte Quellnetze.
- Der Hinweg funktioniert, der Rückweg verwendet eine falsche Route.
- Ein Hostname verweist noch auf eine alte virtuelle IP-Adresse.

Firewalls dürfen nicht pauschal deaktiviert werden. Stattdessen müssen Protokoll, Quelladresse, Zieladresse, Port, Richtung und Treffer der konkreten Regel geprüft werden.

7.9.20 Authentifizierung und Autorisierung abgrenzen

Wenn eine Anwendung eine Anmeldemaske, einen HTTP-Statuscode oder eine konkrete Berechtigungsfehlermeldung zurückgibt, wurde die Anwendungsebene bereits erreicht.

Zu unterscheiden sind:

- **Authentifizierung:** Ist die angegebene Identität gültig?
- **Autorisierung:** Darf diese Identität die gewünschte Ressource verwenden?
- **Anwendungszustand:** Ist Konto, Mandant oder Lizenz in der Anwendung aktiv?
- **Verzeichnisdienst:** Kann die Anwendung LDAP, Active Directory oder einen Identitätsanbieter erreichen?
- **Sitzung:** Sind Cookie, Token oder Ticket gültig?
- **Zeit:** Sind Client, Server und Identitätsdienst ausreichend synchronisiert?

Geeignete Kreuztests:

Test	Erkenntnis
anderer Benutzer am selben Client	benutzerbezogener Fehler möglich

Test	Erkenntnis
gleicher Benutzer an anderem Client	clientbezogener Fehler möglich
kontrolliertes Testkonto	Konto- oder Rechteproblem eingrenzen
anonymer oder öffentlicher Endpunkt	Netzwerk und Anwendung ohne Benutzeranmeldung prüfen
lokales Anwendungskonto	Abhängigkeit vom Verzeichnisdienst untersuchen

Kontosperren und produktive Benutzerkonten müssen berücksichtigt werden. Die Domänenanmeldung wird ausführlich auf Seite 7.10 behandelt.

7.9.21 Anwendungsabhängigkeiten untersuchen

Ein erreichbarer Frontend-Port bedeutet nicht, dass alle Backenddienste funktionieren.

Mögliche Abhängigkeiten:

- Datenbank;
- Verzeichnisdienst;
- DNS;
- API;
- Dateifreigabe;
- Objektspeicher;
- Nachrichtenwarteschlange;
- Cache;
- Lizenzserver;
- Identitätsanbieter;
- externer Cloud-Dienst;
- anderer Container oder interner Dienst.

Typische Befunde:

- Reverse Proxy antwortet mit `502 Bad Gateway`.
- Anwendung antwortet mit `503 Service Unavailable`.
- Anmeldung bleibt hängen, weil der Identitätsdienst nicht erreichbar ist.
- Startseite funktioniert, Datenabfrage schlägt jedoch fehl.
- Healthcheck ist erfolgreich, Geschäftsprozess scheitert.
- Dienst nimmt TCP-Verbindungen an, besitzt aber keine freien Worker.
- Datenträger oder Dateisystem ist voll.
- Container ist erreichbar, aber eine interne Abhängigkeit fehlt.

Anwendungs-, Proxy- und Backendprotokolle müssen anhand des gleichen Fehlerzeitpunkts korreliert werden.

Container-Neustartschleifen werden auf Seite 7.12 und volle Datenträger auf Seite 7.13 ausführlich behandelt.

7.9.22 UDP-Dienste korrekt prüfen

UDP besitzt keinen TCP-ähnlichen Verbindungsaufbau. Ein allgemeiner UDP-Porttest kann daher nicht sicher beweisen, dass der Dienst funktioniert.

Für UDP muss eine gültige Anfrage des tatsächlichen Anwendungsprotokolls gesendet und die Antwort geprüft werden.

DNS-Test unter Windows:

```
Resolve-DnsName `
  -Name <Abzufragender-Name> `
  -Server <DNS-Server>
```

DNS-Test unter Linux oder macOS:

```
dig @<DNS-Server> <Abzufragender-Name> A
```

Ein fehlender UDP-Fehler beweist nicht, dass:

- der Port geöffnet ist;
- der Dienst läuft;
- die Anfrage den Server erreicht hat;
- der Server eine gültige Antwort senden kann;
- der Rückweg funktioniert.

Bei UDP-Diensten sind Anwendungsprotokoll, Serverprotokoll und gegebenenfalls ein kontrollierter Paketmitschnitt besonders wichtig.

7.9.23 IPv4 und IPv6 getrennt prüfen

Ein Name kann gleichzeitig A- und AAAA-Einträge besitzen. Ping und Anwendung können unterschiedliche Adressfamilien auswählen.

Mit curl getrennt testen:

```
curl -4 -v \
  "https://<Servername>:<Port>/<Pfad>"
```

```
curl -6 -v \
  "https://<Servername>:<Port>/<Pfad>"
```

Zu prüfen sind:

- besitzt der Name A- und AAAA-Einträge?
- welche Adresse verwendet die fehlerhafte Anwendung?
- ist der Dienst an IPv4 und IPv6 gebunden?
- besitzen beide Protokolle einen funktionierenden Rückweg?
- gelten für IPv4 und IPv6 unterschiedliche Firewallregeln?
- funktioniert der Proxy oder Load Balancer mit beiden Adressfamilien?
- wird eine veraltete oder nicht erreichbare IPv6-Adresse veröffentlicht?

Wenn ausschließlich IPv6 fehlschlägt, muss der IPv6-Pfad oder der AAAA-Eintrag korrigiert werden. IPv6 sollte nicht ohne Ursachenanalyse dauerhaft deaktiviert werden.

7.9.24 Paketgröße und Übertragungsverhalten berücksichtigen

Ein kleiner ICMP-Ping kann funktionieren, während größere Anwendungsdaten hängen bleiben.

Mögliche Ursachen:

- fehlerhafte Path-MTU-Ermittlung;
- blockierte erforderliche ICMP-Fehlermeldungen;
- Fragmentierungsproblem;
- Paketverlust;
- fehlerhafter VPN-Tunnel;
- Überlastung;
- asymmetrischer Pfad;
- fehlerhafte Netzwerkhardware.

Typische Hinweise:

- TCP-Verbindung wird aufgebaut, aber TLS bleibt während des Handshakes hängen.
- Kleine Webseiten funktionieren, Downloads brechen ab.
- Anwendung funktioniert ohne VPN, aber nicht über den Tunnel.
- Wiederholte TCP-Übertragungen sind im Paketmitschnitt sichtbar.
- Fehler tritt erst ab einer bestimmten Datenmenge auf.

Ein erfolgreicher Standard-Ping darf deshalb nicht als vollständiger Nachweis eines fehlerfreien Datenpfads verwendet werden.

7.9.25 Paketmitschnitt kontrolliert einsetzen

Linux oder macOS:

```
sudo tcpdump -ni <Interface> \
  host <IP-Adresse> and port <Port>
```

Alternativ kann Wireshark mit einem passenden Erfassungsfilter verwendet werden.

Typische Beobachtungen:

Beobachtung	Einordnung
wiederholte SYN-Pakete ohne Antwort	keine TCP-Antwort; Pfad, Regel oder Ziel prüfen
SYN wird mit RST beantwortet	Port geschlossen oder Verbindung aktiv abgelehnt
SYN, SYN-ACK und ACK sichtbar	TCP-Verbindung wurde aufgebaut
TCP-Verbindung wird danach sofort geschlossen	Dienst oder Protokoll lehnt Verbindung ab
TLS ClientHello ohne passende Fortsetzung	TLS-Pfad, Server oder Inspektion prüfen
TLS-Alert	konkrete TLS-Ursache auswerten
viele Wiederholungen	Paketverlust, Überlastung oder fehlerhafter Pfad möglich
Antwort verlässt den Server, erreicht Client aber nicht	Rückweg oder zwischengeschaltete Regel prüfen

Ein Paketmitschnitt beweist nur, was an der jeweiligen Erfassungsstelle sichtbar ist. Bei komplexen Pfaden können Mitschnitte auf Client, Server und einem zwischengeschalteten System erforderlich sein.

Paketmitschnitte dürfen nur mit entsprechender Berechtigung erstellt und müssen datenschutzgerecht gespeichert werden.

7.9.26 Systematischer Diagnoseablauf

1. **Fehler aufnehmen**
   Anwendung, Benutzer, Client, Zeitpunkt, Meldung und ursprüngliche Benutzerfunktion dokumentieren.
2. **Umfang bestimmen**
   Einen Benutzer, einen Client, ein Netz oder alle Benutzer unterscheiden.
3. **Exaktes Ziel bestimmen**
   Servername, Adresse, Port, Protokoll, Pfad und Verbindungsweg erfassen.
4. **Namensauflösung prüfen**
   A- und AAAA-Antworten sowie tatsächlich verwendete Adresse vergleichen.
5. **Ping korrekt bewerten**
   Nur ICMP-Erreichbarkeit als bestätigt betrachten.
6. **Transportprotokoll bestimmen**
   TCP und UDP unterscheiden.

7. **TCP-Port oder UDP-Anwendung prüfen**
   Einen zum tatsächlichen Protokoll passenden Test verwenden.
8. **Listener kontrollieren**
   Port, Prozess und Bindungsadresse auf dem Server prüfen.
9. **Lokalen Anwendungstest durchführen**
   Dienst auf dem Server über den vorgesehenen Namen, Port und Pfad testen.
10. **Externen Anwendungstest durchführen**
    Denselben Dienst vom betroffenen Client aus prüfen.
11. **TLS untersuchen**
    SNI, Zertifikat, Vertrauenskette und Protokollkompatibilität prüfen.
12. **Proxy und VPN berücksichtigen**
    Tatsächlichen Verbindungsweg der Anwendung nachvollziehen.
13. **Anwendungsantwort auswerten**
    Statuscode, Protokollmeldung, Umleitung oder Authentifizierungsfehler bestimmen.
14. **Abhängigkeiten prüfen**
    Datenbank, Identitätsdienst, API, Speicher oder andere Backends untersuchen.
15. **Protokolle korrelieren**
    Client-, Server-, Proxy- und Anwendungsprotokolle auf denselben Zeitraum begrenzen.
16. **Bei Bedarf Paketverlauf erfassen**
    SYN, RST, TLS-Alert, Wiederholungen und Rückweg untersuchen.
17. **Hypothese formulieren**
    Erwartetes Prüfergebnis und möglichen Gegenbeweis festlegen.
18. **Eine kontrollierte Änderung durchführen**
    Risiko, Rückweg und Erfolgskriterium dokumentieren.
19. **Vollständige Funktion prüfen**
    Nicht nur Ping oder Port, sondern den ursprünglichen Benutzerablauf testen.
20. **Nachkontrolle durchführen**
    Protokolle, Überwachung und andere Benutzerfunktionen prüfen.

7.9.27 Befundmatrix

Befund	Mögliche Erklärung	Nächster Nachweis
Ping auf IP funktioniert, Name nicht	Namensauflösungsproblem	A-, AAAA-, Hosts- und Resolverdaten prüfen
Ping auf Name verwendet falsche Adresse	veralteter oder falscher Namenseintrag	autoritative und clientseitige Antwort vergleichen
Ping funktioniert, TCP-Port hat Timeout	Portverkehr wird verworfen oder Antwort fehlt	Firewall, Routing und Paketverlauf prüfen
Ping funktioniert, TCP-Port wird abgelehnt	kein Listener oder aktive Ablehnung	Listener und Prozess auf dem Server prüfen
TCP-Port funktioniert, Verbindung wird sofort geschlossen	falsches Protokoll oder Dienst lehnt Sitzung ab	Dienstprotokoll und Serverlogs prüfen
TCP funktioniert, TLS schlägt fehl	Zertifikat, SNI, TLS-Version oder Inspektion	`curl -v` oder `openssl s_client` auswerten

Befund	Mögliche Erklärung	Nächster Nachweis
TLS funktioniert, HTTP 401	Authentifizierung erforderlich oder fehlerhaft	Benutzer-, Token- oder Identitätsanbieter prüfen
TLS funktioniert, HTTP 403	fehlende Autorisierung oder Richtlinie	Rollen und Zugriffsrichtlinie prüfen
HTTP 404	falscher Pfad oder virtuelle Zuordnung	URL, Host-Header und Proxyroute prüfen
HTTP 407	Proxy-Authentifizierung erforderlich	Proxyweg und Benutzerkontext prüfen
HTTP 502	Gateway erreicht Backend nicht korrekt	Reverse-Proxy- und Backendprotokolle prüfen
HTTP 503	Dienst vorübergehend nicht verfügbar	Healthcheck, Kapazität und Abhängigkeiten prüfen
lokal funktioniert, entfernt nicht	Bindung, Hostfirewall oder Netzwerkpfad	Listeneradresse und Regeln vergleichen
`curl` funktioniert, Anwendung nicht	anwendungsspezifische Konfiguration	Proxy, Cache, Zertifikatsspeicher und Benutzerkontext prüfen
nur ein Client betroffen	lokaler Clientfehler	anderen Client und gleichen Benutzer testen
nur ein Benutzer betroffen	Konto, Profil, Rechte oder Sitzung	anderes Konto am gleichen Client testen
nur über VPN betroffen	Route, DNS, MTU oder VPN-Regel	mit und ohne VPN vergleichen
nur IPv6 betroffen	AAAA-, IPv6-Routing- oder Firewallfehler	`curl -4` und `curl -6` vergleichen
kleine Anfragen funktionieren	MTU, Paketverlust oder Kapazität möglich	Paketverlauf und größere Übertragung prüfen
Port erreichbar, Geschäftsprozess scheitert	Backend oder Anwendungslogik fehlerhaft	vollständigen Ablauf und Abhängigkeiten prüfen

---

7.9.28 Mögliche Ursachen und erforderliche Nachweise

Mögliche Ursache	Erforderlicher Nachweis
falscher Port	Anwendungskonfiguration und tatsächlicher Listener zeigen unterschiedliche Ports
Dienst gestoppt	kein Listener vorhanden und Dienststatus beziehungsweise Logs bestätigen den Ausfall
falsche Listenerbindung	Dienst lauscht nur auf Loopback oder einer anderen Adresse
Hostfirewall blockiert Port	konkrete Regel oder Protokoll zeigt den verworfenen Verbindungsversuch
Netzwerkfirewall oder ACL	Trefferprotokoll oder beidseitiger Paketmitschnitt bestätigt die Unterbrechung

Mögliche Ursache	Erforderlicher Nachweis
falscher DNS-Eintrag	Anwendung verwendet eine nachweislich falsche oder veraltete Adresse
fehlerhafter IPv6-Pfad	IPv6-Test scheitert reproduzierbar, IPv4-Test funktioniert
Proxyfehler	Anwendung verwendet einen fehlerhaften Proxyweg, während direkter Test anders reagiert
Reverse-Proxy-Fehler	Proxy antwortet, passende Route oder gesundes Backend fehlt
Load-Balancer-Fehler	virtuelle Adresse ist erreichbar, aber Healthcheck oder Backendzustand ist fehlerhaft
TLS-Fehler	TLS-Ausgabe zeigt Zertifikats-, SNI- oder Protokollproblem
fehlende Authentifizierung	Anwendung antwortet mit konkretem Anmelde- oder Tokenfehler
fehlende Autorisierung	Anmeldung funktioniert, Ressourcenzugriff wird nachvollziehbar verweigert
ausgefallene Backendabhängigkeit	Frontend antwortet, Serverlogs belegen den Fehler der Abhängigkeit
Ressourcenengpass	Fehler korreliert mit CPU, Speicher, Worker-, Verbindungs- oder Datenträgerengpass
Path-MTU- oder Paketverlustproblem	Verbindung beginnt, größere Übertragung scheitert und Paketverlauf bestätigt Wiederholungen oder Größenproblem
clientseitiger Anwendungsfehler	gleicher Benutzer und Dienst funktionieren mit einem anderen Client
benutzerbezogener Fehler	anderer Benutzer arbeitet am gleichen Client erfolgreich

7.9.29 Kontrollierte Maßnahmen, Risiko und Rückweg

Maßnahme	Risiko	Rückweg
falschen Servernamen oder Port korrigieren	Verbindung erreicht anderes Ziel	ursprüngliche Konfiguration wiederherstellen
gestoppten Dienst kontrolliert starten	Dienst kann erneut fehlschlagen oder Last erzeugen	Dienstzustand und Starttyp dokumentieren
Listenerbindung korrigieren	Dienst wird auf zusätzlichen Netzen erreichbar	vorherige Bindung wiederherstellen
minimale Firewallfreigabe ergänzen	zusätzliche Erreichbarkeit	konkrete Regel entfernen oder deaktivieren
falschen DNS-Eintrag korrigieren	Clients wechseln auf neues Ziel	vorherigen Wert und TTL dokumentieren

Maßnahme	Risiko	Rückweg
Reverse-Proxy-Route korrigieren	andere Anwendungen können betroffen sein	vorherige Proxykonfiguration zurückspielen
fehlerhaftes Backend aus dem Load Balancer nehmen	geringere Kapazität	Backend nach erfolgreicher Prüfung wieder aufnehmen
Proxykonfiguration korrigieren	anderer Netzwerkpfad wird verwendet	ursprüngliche Proxywerte wiederherstellen
Zertifikatskette korrigieren	TLS-Dienst muss eventuell neu geladen werden	vorherige Zertifikatskonfiguration sichern
Anwendungscache kontrolliert leeren	Sitzungs- oder Anmeldedaten können verloren gehen	Benutzer informieren und Ausgangszustand dokumentieren
Backenddienst wiederherstellen	abhängige Anwendungen können beeinflusst werden	dienstspezifischen Wiederherstellungsplan verwenden
Ressource freigeben oder Kapazität erhöhen	Lastverteilung kann sich ändern	vorherige Kapazitäts- oder Ressourcenwerte sichern

Pro Maßnahme sollte möglichst nur eine relevante Variable verändert werden.

7.9.30 Verifikation

Nach einer Maßnahme müssen mindestens folgende Prüfungen erfolgen:

- Servername wird in die erwartete Adresse aufgelöst;
- richtige IPv4- oder IPv6-Adresse wird verwendet;
- vorgesehener TCP- oder UDP-Dienst ist erreichbar;
- richtiger Prozess lauscht auf dem vorgesehenen Port;
- TLS-Handshake und Zertifikatsprüfung funktionieren;
- Anwendung liefert die erwartete Protokollantwort;
- Anmeldung funktioniert;
- benötigte Ressource kann geöffnet werden;
- vollständiger ursprünglicher Benutzerablauf funktioniert;
- Frontend und Backend arbeiten zusammen;
- Test funktioniert aus dem betroffenen Netz;
- bei größerem Umfang funktionieren mehrere repräsentative Clients;
- keine neuen Fehler erscheinen in Client-, Server- oder Proxyprotokollen;
- temporäre Freigaben oder Diagnoseänderungen wurden zurückgenommen;
- Firewall und Zertifikatsprüfung sind weiterhin aktiv;
- Überwachung prüft nicht nur Ping, sondern auch die Anwendungsfunktion;
- keine andere Anwendung wurde durch die Änderung beeinträchtigt.

Ein erfolgreicher Ping oder Porttest allein ist keine ausreichende Verifikation.

7.9.31 Präventionsmaßnahmen

- Anwendungserreichbarkeit zusätzlich zu ICMP überwachen;
- TCP-Port, TLS und einen geeigneten Anwendungsendpunkt prüfen;
- kritische Geschäftsprozesse mit synthetischen Funktionstests überwachen;
- Servername, Port, Protokoll und Abhängigkeiten dokumentieren;
- DNS-Änderungen mit TTL und Rückweg planen;
- Zertifikatsablauf und vollständige Zertifikatskette überwachen;
- Proxy-, VPN- und Firewallregeln dokumentieren;
- Load-Balancer-Healthchecks an die tatsächliche Dienstfunktion anpassen;
- Listenerbindungen und Portzuordnungen dokumentieren;
- IPv4 und IPv6 getrennt überwachen;
- Kapazitätsgrenzen für Worker, Verbindungen, CPU, Speicher und Datenträger überwachen;
- Frontend- und Backendprotokolle zeitlich synchronisieren;
- Änderungen an DNS, Firewall, Proxy und Anwendung nachvollziehbar protokollieren;
- Wiederherstellungs- und Rückfallverfahren testen;
- Monitoring nicht auf „Host ist pingbar“ beschränken.

7.9.32 Typische Fehler bei der Diagnose

- Einen erfolgreichen Ping mit einer funktionierenden Anwendung gleichsetzen.
- Einen Ping als Test eines TCP- oder UDP-Ports interpretieren.
- Einen anderen Port als den tatsächlich konfigurierten Port testen.
- Die IP-Adresse statt des vorgesehenen HTTPS-Namens aufrufen.
- SNI, Host-Header und Zertifikatsnamen ignorieren.
- `curl` für einen nicht unterstützten oder nicht HTTP-basierten Dienst verwenden.
- Einen allgemeinen UDP-Test als sicheren Funktionsnachweis betrachten.
- Einen Timeout automatisch als Firewallfehler einstufen.
- `Connection refused` und Timeout gleichsetzen.
- Nur einen lokalen Test auf dem Server durchführen.
- Nur prüfen, ob der Prozess läuft, statt den Listener zu kontrollieren.
- Nur prüfen, ob der Port offen ist, statt das Anwendungsprotokoll zu testen.
- Eine HTTP-Antwort automatisch als vollständigen Anwendungserfolg werten.
- Einen Healthcheck mit dem eigentlichen Geschäftsprozess gleichsetzen.
- Proxy, VPN, Load Balancer oder Reverse Proxy ignorieren.
- IPv4 und IPv6 nicht getrennt prüfen.
- Firewalls vollständig deaktivieren.
- Zertifikatsprüfungen dauerhaft umgehen.
- Vor der Protokollsicherung Dienste oder Server neu starten.
- Zugangsdaten, Token oder vollständige Debugausgaben ungeschützt weitergeben.
- Nach einer Änderung nur erneut pingen und die ursprüngliche Benutzerfunktion nicht testen.

7.9.33 Typische Prüfungsfragen

Warum beweist ein erfolgreicher Ping keine funktionierende Anwendung?

Ping prüft ICMP-Echo-Kommunikation. Die Anwendung kann einen anderen Transportweg, einen bestimmten TCP- oder UDP-Port, TLS, Authentifizierung und weitere Backenddienste benötigen.

Welcher Test sollte nach einem erfolgreichen Ping durchgeführt werden?

Zuerst müssen Zielname, Protokoll und Port der Anwendung bestimmt werden. Anschließend wird der konkrete TCP-Port oder das tatsächliche UDP-Anwendungsprotokoll geprüft.

Was bedeutet „Connection refused“?

Die TCP-Verbindung wurde aktiv abgelehnt. Häufig fehlt ein Listener auf dem Zielport oder eine Komponente lehnt die Verbindung gezielt ab. Der genaue Absender der Ablehnung muss bei Bedarf mit Protokollen oder einem Paketmitschnitt bestimmt werden.

Was bedeutet ein TCP-Timeout?

Der Verbindungsaufbau wurde nicht rechtzeitig abgeschlossen. Mögliche Ursachen sind eine verwerfende Firewallregel, ein falscher Netzwerkpfad, ein fehlender Rückweg, Überlastung oder eine falsche Zieladresse.

Warum sollte ein HTTPS-Dienst nicht nur über seine IP-Adresse getestet werden?

Virtuelle Hosts, TLS-SNI und Zertifikatsprüfung verwenden den Servernamen. Ein Aufruf über die IP-Adresse kann deshalb einen anderen Dienst oder ein anderes Zertifikat erreichen.

Was beweist ein erfolgreicher TCP-Porttest?

Er beweist, dass zum Testzeitpunkt eine TCP-Verbindung zur geprüften Adresse und zum geprüften Port aufgebaut werden konnte. Die Funktion des Anwendungsprotokolls ist damit noch nicht bewiesen.

Was bedeutet ein HTTP-Statuscode 401 oder 403 für die Netzwerkdiagnose?

Die Anwendung wurde erreicht und hat auf HTTP-Ebene geantwortet. Der Fehler liegt anschließend wahrscheinlich bei Authentifizierung, Autorisierung oder einer Anwendungsrichtlinie.

Warum ist ein allgemeiner UDP-Porttest nicht eindeutig?

UDP besitzt keinen verbindlichen Verbindungsaufbau. Ohne gültige Anwendungsanfrage und auswertbare Antwort lässt sich die Dienstfunktion nicht sicher bestätigen.

Was bedeutet es, wenn die Anwendung lokal auf dem Server funktioniert, vom Client aber nicht?

Der Dienst selbst arbeitet grundsätzlich. Danach müssen Listenerbindung, Hostfirewall, Netzwerkpfad, Proxy, Load Balancer und externe Namensauflösung geprüft werden.

Warum kann `curl` funktionieren, obwohl die eigentliche Anwendung fehlschlägt?

Die Anwendung kann einen anderen Proxy, Zertifikatsspeicher, DNS-Cache, Benutzerkontext, Authentifizierungsmechanismus oder zusätzliche Protokollfunktionen verwenden.

Warum reicht ein erfolgreicher Healthcheck nicht immer aus?

Ein Healthcheck kann nur einen einfachen Endpunkt prüfen. Datenbankzugriff, Anmeldung oder der eigentliche Geschäftsprozess können trotzdem fehlschlagen.

7.9.34 Checkliste

- [ ] ursprüngliche Benutzerfunktion dokumentiert
- [ ] genaue Fehlermeldung und Fehlerzeitpunkt erfasst
- [ ] betroffene Benutzer und Clients bestimmt
- [ ] vollständiger Servername erfasst
- [ ] verwendete Zieladresse bestätigt
- [ ] TCP oder UDP bestimmt
- [ ] richtiger Port bestimmt
- [ ] Protokoll und URL-Pfad bestimmt
- [ ] Ping nur als ICMP-Test bewertet
- [ ] A- und AAAA-Auflösung geprüft
- [ ] tatsächlich verwendete Adresse kontrolliert
- [ ] TCP-Port oder UDP-Anwendung geprüft
- [ ] Listener und zugehöriger Prozess geprüft
- [ ] Bindungsadresse kontrolliert
- [ ] lokaler Anwendungstest durchgeführt
- [ ] entfernter Anwendungstest durchgeführt
- [ ] TLS und Zertifikat geprüft
- [ ] Proxy- und VPN-Weg geprüft
- [ ] Firewall, ACL und Rückweg berücksichtigt
- [ ] Load Balancer oder Reverse Proxy geprüft
- [ ] Anwendungsantwort und Statuscode ausgewertet
- [ ] Authentifizierung und Autorisierung unterschieden
- [ ] Backendabhängigkeiten geprüft
- [ ] IPv4 und IPv6 getrennt getestet
- [ ] Client-, Server- und Proxyprotokolle korreliert

- [ ] Paketmitschnitt nur bei Bedarf und berechtigt erstellt
- [ ] Hypothese und Gegenbeweis formuliert
- [ ] Risiko und Rückweg vor Änderung dokumentiert
- [ ] nur eine kontrollierte Änderung durchgeführt
- [ ] vollständiger Benutzerablauf verifiziert
- [ ] temporäre Diagnoseänderungen zurückgenommen
- [ ] Ergebnis und Präventionsmaßnahme dokumentiert

7.9.35 Schnellreferenz

Aufgabe	Windows	Linux	macOS
Ping	`ping <Server>`	`ping -c 4 <Server>`	`ping -c 4 <Server>`
Namensauflösung	`Resolve-DnsName <Server>`	`getent ahosts <Server>`	`dscacheutil -q host -a name <Server>`
A-Eintrag	`Resolve-DnsName <Server> -Type A`	`dig <Server> A`	`dig <Server> A`
AAAA-Eintrag	`Resolve-DnsName <Server> -Type AAAA`	`dig <Server> AAAA`	`dig <Server> AAAA`
TCP-Port	`Test-NetConnection <Server> -Port <Port>`	`nc -vz <Server> <Port>`	`nc -vz <Server> <Port>`
HTTP oder HTTPS	`curl.exe -v <URL>`	`curl -v <URL>`	`curl -v <URL>`
IPv4 mit curl	`curl.exe -4 -v <URL>`	`curl -4 -v <URL>`	`curl -4 -v <URL>`
IPv6 mit curl	`curl.exe -6 -v <URL>`	`curl -6 -v <URL>`	`curl -6 -v <URL>`
bestimmte Ziel-IP	`curl.exe --resolve <Name>:<Port>:<IP> <URL>`	`curl --resolve <Name>:<Port>:<IP> <URL>`	`curl --resolve <Name>:<Port>:<IP> <URL>`
Listener	`Get-NetTCPConnection -State Listen`	`sudo ss -lntp`	`sudo lsof -nP -iTCP -sTCP:LISTEN`
WinHTTP-Proxy	`netsh winhttp show proxy`	nicht zutreffend	nicht zutreffend
Systemproxy	Anwendungseinstellungen prüfen	Anwendung und Umgebungsvariablen prüfen	`scutil --proxy`
TLS	`openssl s_client` bei installierter OpenSSL-Version	`openssl s_client`	`openssl s_client`
Paketmitschnitt	Wireshark oder freigegebenes Windows-Werkzeug	`tcpdump` oder Wireshark	`tcpdump` oder Wireshark

---

7.9.36 Quellen

Standards und RFCs

- RFC 792 – Internet Control Message Protocol
- RFC 9293 – Transmission Control Protocol
- RFC 9110 – HTTP Semantics
- RFC 8446 – The Transport Layer Security Protocol Version 1.3
- RFC 1191 – Path MTU Discovery für IPv4
- RFC 8201 – Path MTU Discovery für IPv6
- IANA – HTTP Status Code Registry

Offizielle Hersteller- und Projektdokumentation

- Microsoft Learn – Test-NetConnection
- Microsoft Learn – Resolve-DnsName
- Microsoft Learn – Get-NetTCPConnection
- Microsoft Learn – netsh winhttp
- curl – offizielle Befehlsreferenz
- Everything curl – Verbose output
- OpenSSL – openssl s_client
- OpenBSD – nc(1)

Lokale Befehlsreferenzen

Die genaue Syntax kann von Betriebssystem und installierter Version abhängen. Maßgeblich ist die lokale Befehlsreferenz:

```
man ping
man nc
man curl
man ss
man lsof
man tcpdump
man dscacheutil
man scutil
```

Für diese Seite wurden keine Community-Berichte, Hersteller-Social-Media-Aussagen oder eigenen Laborergebnisse als Nachweis verwendet.
