CERTYX

AUTONOMOUS CERTIFICATE LIFECYCLE MANAGEMENT

# Agenteninstallation

CLM-Netzwerkagent — Installation und Erkennung (Discovery)

| SPRACHE | REVISION | DATUM |
| --- | --- | --- |
| Deutsch | 8550060f | 1. November 2026 |

# Inhalt

1 Funktionsweise 3
2 Vor der Installation 3
3 Agent-Adresse und lokaler Testserver 3
4 Fünf Schritte 3
5 Paketprüfung 4
6 Paket-Downloads 4
7 Linux mit systemd 5
8 Windows mit Aufgabenplanung 10
9 Manuelle Ausführung / Ausführung im Container 12
10 Abdeckung und Fortschritt 12
11 Netzwerkkarte 14
12 Für das Sicherheitsteam des Kunden 14
13 Betriebseinstellungen 15
14 Widerruf und Wiederherstellung 15
15 Paketerstellung 16
16 SSH 17
17 Vom Kunden kontrollierte Geheimnisse und KRL-Abonnements 18
18 SSH-Benutzeridentitäten für Workloads 18
19 Ziele für TLS-Client-Zertifikate 18
20 Bereitstellungs-Connectoren und SSH-Relay 19
21 Zustellung gespeicherter Lifecycle-Ergebnisse 22
22 Geschütztes Lesen der Konfiguration 22
23 Unabhängige Verifizierung im privaten Netzwerk 22
24 Öffentliche Zertifizierungsstellen: DNS-01, EAB und DigiCert 23
25 Signierte Updates 24
26 Prüfungen der Befehlszeile 25
27 Voraussetzungen des Windows-Hosts 26

# 1 Funktionsweise

Der Netzwerkagent läuft innerhalb eines Kundennetzwerks. Er baut ausgehende Verbindungen zum HTTPS-Endpunkt von CLM auf; CLM benötigt keinen eingehenden Zugriff auf private Subnetze. Ein registrierter Agent meldet die privaten IPv4-Netze, mit denen sein Host verbunden ist oder zu denen er routet (Netzwerkkarte), ruft sein genehmigtes Discovery-Profil ab, ermittelt aktive Hosts in den genehmigten Bereichen, verbindet sich per TCP-Verbindungsaufbau mit anschließendem direktem TLS-Handshake mit deren TLS-Ports und lädt öffentliche Zertifikate in das Inventar hoch. Private Schlüssel lädt er niemals hoch. Der Scan ist transparent: Er wird vorab mit dem Sicherheitsteam des Kunden abgestimmt (siehe „Für das Sicherheitsteam des Kunden“) und versucht nie, Angriffserkennung zu umgehen oder zu täuschen; Taktung, Grenzen pro Host, Zeitfenster und Drosselung schützen nur Ihr Netzwerk und Ihre Hosts vor Last. Verwenden Sie einen Agenten pro erreichbarer Netzwerkzone. Die Installation eines Discovery-Agenten berechtigt nicht zur Bereitstellung von Zertifikaten.

# 2 Vor der Installation

Wählen Sie einen unterstützten Linux-Host mit systemd oder einen Windows-Host mit Aufgabenplanung, der die autorisierten Ziele im Netzwerk erreicht. Verwenden Sie das Paket, das zu Betriebssystem und Architektur passt: Die Befehle der Konsole verwenden `clm-agent-linux-x64.tar.gz` und `clm-agent-win-x64.zip`; install-linux.sh akzeptiert auch aarch64-Hosts mit `clm-agent-linux-arm64.tar.gz`, wenn Ihr Betreiber dieses Paket veröffentlicht (ersetzen Sie in den Befehlen `x64` durch `arm64`). Die Pakete enthalten die .NET-Laufzeit; auf den Zielhosts ist keine SDK-Installation erforderlich. HTTPS muss einen Hostnamen und eine Zertifikatskette einer Zertifizierungsstelle (CA) verwenden, der das Host-Betriebssystem vertraut. Die Agent-Adresse muss HTTPS verwenden, vom Agent-Host erreichbar sein und ein Zertifikat haben, dem der Agent-Host vertraut und dessen Sperrstatus er online prüfen kann; eine Konsole über unverschlüsseltes HTTP auf 127.0.0.1 kann keine Agents bedienen (siehe „Agent-Adresse und lokaler Testserver“). Erlauben Sie ausgehendes HTTPS zu CLM und ausgehendes TCP zu den Ziel-CIDRs; es wird kein neuer eingehender Listener für den Agenten geöffnet.

# 3 Agent-Adresse und lokaler Testserver

Agents verbinden sich mit einer HTTPS-Adresse, der „Adresse für Agents“, die nach „Registrierungscode erstellen“ angezeigt wird. Sie muss vier Bedingungen erfüllen, die das Installationsprogramm prüft, bevor es den Code verwendet (`Clm.Agent check`): Der Agent-Host löst den Namen auf und erreicht den Port; das Serverzertifikat nennt genau diesen Hostnamen oder diese IP-Adresse und führt zu einem Stammzertifikat im System-Zertifikatsspeicher des Hosts; das Zertifikat nennt einen CRL-Verteilungspunkt (oder OCSP-Responder), den der Agent-Host über HTTP abrufen kann; und die Uhren weichen um weniger als vier Minuten ab. Der Betreiber konfiguriert sie mit `Clm:Ingress:AgentHosts` (erster Eintrag), optional `Clm:Ingress:AgentPort`, wenn Agents einen anderen Port als die Konsole verwenden, und `Clm:Ingress:AgentTrustRootFile`, wenn das Zertifikat von einer privaten CA stammt: Die Konsole veröffentlicht dieses Stammzertifikat dann unter `/downloads/clm-agent-trust-root.crt` und zeigt die Befehle, um seine SHA-256-Prüfsumme zu prüfen und ihm zu vertrauen (siehe CONFIGURATION.md). Ohne Agent-Host nennt die Konsole die HTTPS-Adresse, über die sie geöffnet wurde; über einen Proxy geöffnet (etwa den Edge der Container-Bereitstellung), über den sich Agents nicht authentifizieren können, nennt sie diesen Hostnamen mit `Clm:Ingress:AgentPort` (8443 in der Container-Bereitstellung, die TLS unverändert an den Agent-Listener weiterreicht), ohne Agent-Port keine; über unverschlüsseltes HTTP geöffnet nennt sie keine. In beiden Fällen erklärt sie den Grund.

**Lokaler Testserver.** `./start.sh` stellt die Konsole nur auf diesem Computer bereit (https://127.0.0.1:7243 mit dem Entwicklungszertifikat, http://127.0.0.1:5180 leitet dorthin um); Agents auf anderen Rechnern können diese Adresse weder erreichen noch ihr vertrauen. Starten Sie es mit der Adresse dieses Computers, die der Agent-Host erreicht (seine LAN-Adresse, ein DNS-Name oder 172.17.0.1 für Docker-Container auf demselben Computer):

```
ip -4 addr          # oder: hostname -I  - die vom Agent-Host erreichbare Adresse wählen
CLM_AGENT_HOST=192.168.1.20 ./start.sh
```

start.sh erstellt dann in `.local-agent-endpoint/` eine lokale Stamm-CA und ein HTTPS-Zertifikat für diese Adresse (nur für den Besitzer, nie eingecheckt; der CA-Schlüssel bleibt dort), fügt einen HTTPS-Listener für Agents auf Port 5443 hinzu (`CLM_AGENT_PORT`), veröffentlicht die Sperrliste über HTTP auf Port 5181 (`CLM_AGENT_CRL_PORT`) und konfiguriert Agent-Host, Port und Stammzertifikat. Es gibt die Agent-Adresse aus; die Konsole nennt sie zusammen mit dem Vertrauensschritt. Die Adresse wird für das nächste `./start.sh` gespeichert; `CLM_AGENT_HOST=off ./start.sh` schaltet sie ab. Erlauben Sie in der Firewall dieses Computers eingehendes TCP 5443 und 5181 (zum Beispiel `sudo ufw allow 5443/tcp` und `sudo ufw allow 5181/tcp`). Die Konsole selbst bleibt nur unter 127.0.0.1 und localhost erreichbar. Dies dient Tests; im Produktivbetrieb verwenden Sie einen Hostnamen mit einem Zertifikat Ihrer PKI oder einer öffentlichen CA.

# 4 Fünf Schritte

1. Geben Sie unter Agenten und Ziele eine eindeutige Agenten-ID ein und wählen Sie „Registrierungscode erstellen“. Die Konsole zeigt dann die Adresse für Agents, Ihre Organisations-ID und nummerierte Installationsschritte mit kopierfertigen Befehlen für Linux und Windows. Laden Sie die Datei mit dem Registrierungstoken herunter und kopieren Sie sie vor allem anderen in ein leeres Arbeitsverzeichnis auf dem Server: Das Paketarchiv wird nur mit einem gültigen Registrierungscode ausgeliefert, daher sendet der Paket-Download den Code aus dieser Datei. Sie läuft nach zehn Minuten ab und kann nur einmal verwendet werden; die Prüfung beim Download verbraucht den Code nicht, erst die Registrierung. Behandeln Sie sie wie ein Passwort; übergeben Sie ihren Inhalt niemals als Befehlsargument.
2. Laden Sie im Arbeitsverzeichnis das Paket herunter und prüfen Sie seine SHA-256-Prüfsumme (Schritt 3 im Abschnitt zu Linux oder Windows). Wenn Ihr Betreiber einen Release-Signaturschlüssel veröffentlicht, prüfen Sie zusätzlich das signierte Release: Laden Sie `clm-agent-<rid>.release.json`, `clm-agent-<rid>.release.json.sig` und `verify-release.sh` von derselben `/downloads/`-Adresse herunter (`<rid>` ist `linux-x64` oder `win-x64`; eine separate

Agent-Adresse liefert diese Dateien, die beiden Paketprüfer und `verify-release.sh` ebenfalls), beziehen Sie release-signing.pub und seinen SHA-256-Fingerabdruck über einen vertrauenswürdigen Kanal (nicht nur vom selben Server) und führen Sie vor dem Entpacken `bash verify-release.sh release-signing.pub clm-agent-linux-x64.release.json clm-agent-linux-x64.tar.gz` aus; installieren Sie nicht, wenn die Prüfung fehlschlägt. Ohne veröffentlichten Schlüssel gibt es keine `.sig`-Datei, und diese Signaturprüfung entfällt; solche Releases können nicht als signierte Updates installiert werden. Prüfen Sie danach seine Herausgebersignatur gegen den Fingerabdruck des Herausgeberzertifikats, den Sie über einen unabhängigen Kanal von Ihrem CLM-Anbieter erhalten haben (siehe „Paketprüfung“); installieren Sie kein abgewiesenes Paket. Windows-Pakete tragen bei einem signierten Release zusätzlich Authenticode-Signaturen.

3. Installieren Sie das Paket mit den Befehlen, die die Konsole anzeigt (Schritt für Schritt in den Abschnitten zu Linux und Windows unten). Der Agent erzeugt lokal eine RSA-3072-Client-Identität. CLM speichert ihren öffentlichen Fingerabdruck in der richtigen Organisation. Die geheime PFX-Datei und die Passwortdatei verbleiben auf dem Host.
4. Wählen Sie in CLM den registrierten Agenten aus und genehmigen Sie sein Discovery-Profil: IPv4-CIDRs (die Netzwerkkarte des Agenten hilft dabei; öffentliche Bereiche werden nie vorgeschlagen und müssen vom Administrator eingegeben und bestätigt werden), Ausnahmen, Modus (`standard`: Erreichbarkeit und gängige Ports; `full`: zusätzlich jeder Port 1–65535 auf aktiven Hosts), Erreichbarkeitsports, gängige Ports, Rate, Verbindungen pro Host sowie Scanfenster mit ihrer Zeitzone. Aktivieren Sie das Profil. Der Agent prüft jedes Profil selbst, bevor er Verbindungen aufbaut: höchstens 64 CIDRs, jeweils mindestens /16 und höchstens 65.536 Adressen, höchstens 262.144 Adressen pro Profil und 256 Ausnahmen (oder die strengeren auf dem Server konfigurierten Grenzen), nur wohlgeformte IPv4-CIDRs und -Ausnahmen ohne gesetzte Host-Bits, niemals Loopback-, Link-Local-, Multicast- oder reservierte Adressen; öffentliche Bereiche, Bereiche kürzer als /20 und der Modus `full` nur mit Bestätigung des Administrators. Ein abgelehntes Profil wird als ungültig gemeldet, und es wird nichts gescannt. Die lokale Liste `AllowedCidrs` (anfangs die direkt angeschlossenen IPv4-Netze) ist nur dann eine feste Grenze, wenn der Host-Eigentümer `RestrictToAllowedCidrs` auf `true` setzt.
5. Prüfen Sie das Ergebnis: Das Installationsprogramm endet mit `sudo clm-agent status` („Result: connected“), die Konsole zeigt „erster Kontakt empfangen“ für den neuen Agenten und unter Registrierte Agents seinen ersten und letzten Kontakt. Später prüfen Sie den letzten Kontakt, das Agentenprotokoll und neu erkannte Zertifikate im Inventar. Erkannte Zertifikate haben zunächst den Verantwortlichen „Nicht zugewiesen“. TLS-Fehler werden aufbewahrt; die Erkennung ist kein Nachweis, dass dem Dienstzertifikat vertraut wird.

## 5 Paketprüfung

Jedes Release-Paket trägt zwei unabhängige Signaturen:

- Code-Signierung des Herausgebers (vor der Installation geprüft): `<Archiv>.sha256` ist der Release-Hash und `<Archiv>.sha256.p7s` dessen abgetrennte CMS-Signatur mit dem Code-Signing-Zertifikat des Herausgebers (eine optionale `.p7s.tsr`-Datei enthält einen RFC-3161-Zeitstempel). Im Paket listet `SHA256SUMS` den SHA-256-Hash jeder Datei, `SHA256SUMS.p7s` signiert ihn; fehlende, hinzugefügte oder geänderte Dateien und Links werden abgelehnt. Windows-Pakete tragen zusätzlich Authenticode-Signaturen auf `Clm.Agent.exe`, den `Clm.*.dll`-Assemblys und den PowerShell-Skripten.
- Das offline signierte ECDSA-P-384-Release-Manifest (`.release.json` und `.sig`) für `verify-release.sh` und signierte Linux-Updates (siehe „Signierte Updates“).

Vertrauensanker der Code-Signierung ist der SHA-256-Fingerabdruck des Herausgeberzertifikats. Übernehmen Sie ihn nicht vom Download-Server: Beziehen Sie ihn von Ihrem CLM-Anbieter über einen unabhängigen Kanal (Vertragsunterlagen, Supportportal oder Release Notes) und fixieren Sie ihn. Optional verlangen Sie zusätzlich eine Kette zur CA des Herausgebers mit `--ca-file` / `-CaFile`. Die Prüfung funktioniert offline und benötigt nur bash, OpenSSL 3 und coreutils (Linux) oder Windows PowerShell 5.1 / PowerShell 7 (Windows).

```
bash verify-agent-package.sh --publisher-sha256 FINGERABDRUCK clm-agent-linux-x64.tar.gz
```

```
Get-AuthenticodeSignature .\verify-agent-package.ps1
.\verify-agent-package.ps1 -PublisherSha256 FINGERABDRUCK -Archive .\clm-agent-win-x64.zip
```

`VERIFIED` mit Exit-Code 0 bedeutet, dass Signatur, fixierter Herausgeber, Code-Signing-Verwendung, Zertifikatsgültigkeit und alle Hashes übereinstimmen. `REFUSED: <Grund>` mit Exit-Code 1 bedeutet: nicht installieren (fehlende Signatur, unerwarteter Herausgeber, Hash-Abweichung oder abgelaufenes Zertifikat). Laden Sie das Paket erneut herunter und wenden Sie sich an Ihren Anbieter, wenn die Ablehnung bestehen bleibt. Mit `--tsa-ca-file` (nur im Linux-Prüfer) ist der RFC-3161-Zeitstempel erforderlich, und die Zertifikatsgültigkeit wird zum Signaturzeitpunkt geprüft. `--allow-unsigned-development` / `-AllowUnsignedDevelopment` ist nur für Entwicklungs-Builds gedacht: Es akzeptiert eine fehlende Signatur nach der Hash-Prüfung und gibt eine deutliche Warnung aus. Verwenden Sie es nie auf Produktivhosts. Die Konsole erstellt ihre Installationsbefehle aus dem Paketkatalog: Veröffentlicht der Server ein Paket ohne Herausgebersignatur (keine `.sha256.p7s`, ein unsignierter Entwicklungsbuild), sagen die Befehle das in einer Kommentarzeile, laden keine `.p7s` herunter, prüfen die Prüfsumme und verwenden diese Entwicklungsoptionen; ein signiertes Paket erhält immer die Befehle mit fixiertem Herausgeber. Beide Prüfer prüfen mit `--directory <Verzeichnis>` / `-Directory <Verzeichnis>` auch ein entpacktes Paket (signierte `SHA256SUMS`, jeden Hash und den exakten Dateibestand); genau das führen die Installationsprogramme aus. In diesem Modus prüft der PowerShell-Prüfer unter Windows zusätzlich die Authenticode-Signaturen von `Clm.Agent.exe`, den `Clm.*.dll`-Assemblies und den PowerShell-Skripten gegen denselben fixierten Herausgeber (die Kette muss auf dem Host vertrauenswürdig sein, oder geben Sie `-CaFile` an); unter Linux und macOS gibt er einen Hinweis aus und stützt sich auf das signierte Manifest. Die Prüfung eines Archivs (`-Archive`) prüft nur den signierten Archiv-Hash.

Die Installationsprogramme wiederholen die Prüfung und brechen ab, bevor der Host verändert wird. Da diese Prüfung Code aus dem Paket selbst ausführt, schützt sie vor beschädigten oder teilweise veränderten Paketen und erzwingt Ihren fixierten Fingerabdruck; sie ersetzt nicht die vorherige Prüfung des Archivs mit dem separat heruntergeladenen Prüfer.

## 6 Paket-Downloads

Die Paketarchive ( `/downloads/clm-agent-*.tar.gz` und `.zip` ) enthalten ausführbaren Code und werden nur an eine Anfrage ausgeliefert, die eines davon vorweist:

- eine angemeldete Konsolensitzung: Die Paketkarten unter Agenten und Ziele und in der SSH-Automatisierung funktionieren in Ihrem Browser. Jede Karte zeigt den Dateinamen des Archivs, Version, Größe, SHA-256 (mit Kopieren-Schaltfläche), ob eine Herausgebersignatur ( `.p7s` ) und ein signiertes Release-Manifest ( `.sig` ) veröffentlicht sind, oder dass das Paket auf diesem Server nicht veröffentlicht ist; ihre Schaltflächen laden das Paket, seine Dateien `.sha256` , `.p7s` und `.tsr` , das Release-Manifest mit seiner Signatur und den Prüfer herunter, jeweils unter dem eigenen Dateinamen (der Server sendet jede Datei unter `/downloads/` als Anhang). Die Karte zeigt, welche Signaturdateien vorhanden sind; geprüft werden sie vom Prüfer auf dem Host;
- einen gültigen, unverbrauchten und nicht abgelaufenen Registrierungscode Ihrer Organisation, gesendet in den Headern `X-CLM-Enrollment-Token` und `X-CLM-Tenant` (die Organisations-ID). Die Befehle der Konsole lesen den Code aus clm-enrollment.token im Arbeitsverzeichnis; unter Linux wird er per Pipe an curl übergeben und erscheint nie in einer Befehlszeile. Die Prüfung verbraucht den Code nicht; er wird einmal verbraucht, durch die Registrierung, und die Gültigkeit von zehn Minuten gilt für Download und Installation zusammen;
- das TLS-Client-Zertifikat eines registrierten Agents mit seiner Organisation (signierte Updates, siehe „Signierte Updates“).

Jede andere Anfrage erhält HTTP 401, das curl als `The requested URL returned error: 401` und PowerShell als `(401) Unauthorized` meldet. Das bedeutet: clm-enrollment.token fehlt im aktuellen Verzeichnis, der Code ist abgelaufen oder wurde bereits verwendet, oder die Organisations-ID im Befehl ist falsch. Erstellen Sie einen neuen Code für dieselbe Agenten-ID, kopieren Sie die neue Datei in das Arbeitsverzeichnis und führen Sie die Befehle erneut aus. Prüfsummen ( `.sha256` ), Signaturen ( `.p7s` , `.tsr` ), Release-Manifeste ( `.release.json` , `.sig` ), die Prüfskripte und `clm-agent-trust-root.crt` enthalten keinen Code und bleiben öffentlich, sodass Sie ein Paket ohne Berechtigungsnachweis prüfen können.

## 7 Linux mit systemd

Am schnellsten und sichersten geht es über die Konsole: Nach „Registrierungscode erstellen“ zeigt Agenten und Ziele diese Schritte mit Ihrer echten Adresse und Organisations-ID bereits ausgefüllt und zum Kopieren an. Verwenden Sie die folgenden Befehle nur, wenn Sie sie nicht aus der Konsole kopieren können; ersetzen Sie dann jeden Platzhalter in spitzen Klammern einschließlich der Klammern durch den echten Wert:

- `<CONTROL-PLANE-ORIGIN>` : die HTTPS-Adresse, die Agenten verwenden, zum Beispiel `https://agent.ihre-firma.de` (ohne Pfad, ohne abschließenden Schrägstrich). Die Konsole zeigt sie nach „Registrierungscode erstellen“ als „Adresse für Agents“: den Agenten-Hostnamen, wenn Ihr Betreiber einen konfiguriert hat ( `Clm:Ingress:AgentHosts` ), sonst die HTTPS-Adresse, über die Sie die Konsole öffnen (über einen Proxy mit dem Agent-Port, zum Beispiel `https://certyx.ihre-firma.de:8443` ). `https://clm.example.com` ist nur ein Beispiel; das Installationsprogramm lehnt es ab.
- `<ORGANIZATION-ID>` : Ihre Organisations-ID, an derselben Stelle als „Organisations-ID“ angezeigt.
- `<PUBLISHER-SHA256>` : der SHA-256-Fingerabdruck des Herausgeberzertifikats der Pakete. Beziehen Sie ihn über einen unabhängigen Kanal von Ihrem CLM-Anbieter (Vertragsunterlagen, Supportportal oder Release Notes), nie nur vom Download-Server, und fixieren Sie ihn (siehe „Paketprüfung“). Die Konsole zeigt ihn nicht an.
- `<ADMIN-USER>` und `<SERVER>` : Ihr Login auf dem Linux-Server und dessen Hostname; nur zum Kopieren der Token-Datei in Schritt 2.

## 7.1 Schritt 1: Registrierungscode erstellen




Agenten und Ziele nach „Registrierungscode erstellen“: Adresse für Agenten, Organisations-ID und die nummerierten Schritte mit Befehlen zum Kopieren (Beispieldaten)

Geben Sie in CERTYX → Agenten und Ziele eine Agenten-ID ein (Buchstaben, Ziffern, - und _; eine ID pro Agent), wählen Sie „Registrierungscode erstellen“ und dann „Registrierungstoken-Datei herunterladen“. Nur Administratoren können Codes erstellen; eine bereits registrierte Agenten-ID wird abgewiesen, wählen Sie für einen Ersatz-Agenten daher eine neue ID. Ein weiterer Code für dieselbe, noch nicht registrierte Agenten-ID ist erlaubt; jeder Code funktioniert für sich. Die Datei clm-enrollment.token gilt zehn Minuten und nur einmal: Führen Sie die Schritte 2–4 (Kopieren, Download und Installation) innerhalb von zehn Minuten aus oder erstellen Sie einen neuen Code für dieselbe Agenten-ID. Behandeln Sie sie wie ein Passwort und fügen Sie ihren Inhalt nie in einen Befehl ein.

## 7.2 Schritt 2: Token-Datei in ein Arbeitsverzeichnis kopieren

Legen Sie auf dem Linux-Server ein leeres Arbeitsverzeichnis an und kopieren Sie clm-enrollment.token von dem Computer, auf den Sie sie heruntergeladen haben, in dieses Verzeichnis, zum Beispiel mit `scp clm-enrollment.token <ADMIN-USER>@<SERVER>:` (Ihr Home-Verzeichnis auf dem Server) oder mit WinSCP. Behalten Sie den Dateinamen bei. Führen Sie die Befehle der Schritte 3 und 4 von dort aus.

## 7.3 Schritt 3: Paket herunterladen, prüfen und entpacken

Zeigt die Konsole den Schritt „Dieses CERTYX verwendet für die Agent-Adresse eine eigene Zertifizierungsstelle“ (der Betreiber hat `Clm:Ingress:AgentTrustRootFile` gesetzt, zum Beispiel der lokale Testserver), führen Sie ihn zuerst aus, im Arbeitsverzeichnis und im selben Terminal wie die folgenden Schritte, denn schon der Paket-Download braucht dieses Vertrauen. Die Prüfsumme steht in der Konsole neben dem Befehl; `-k` ist hier nur deshalb sicher, weil die heruntergeladene Datei mit ihr verglichen wird, und `sha256sum` muss `certyx-agent-root.crt: OK` ausgeben. Der dritte Befehl lässt die folgenden `curl`-Downloads dieses Terminals dem Stammzertifikat vertrauen; das Installationsprogramm fügt es anschließend selbst dem System-Zertifikatsspeicher hinzu (`CLM_AGENT_TRUST_ROOT` im Installationsbefehl, siehe Schritt 4), sodass kein Befehl für den Zertifikatsspeicher von Hand eingegeben wird:

```
curl -fsSLk -o certyx-agent-root.crt <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-trust-root.crt
echo "<SHA256-FROM-CONSOLE>  certyx-agent-root.crt" | sha256sum -c
export CURL_CA_BUNDLE="$PWD/certyx-agent-root.crt"
```

Danach auf dem Linux-Server (x64 mit systemd, ausgehendes HTTPS zu CERTYX) als Administrator im Arbeitsverzeichnis mit clm-enrollment.token. Der erste Befehl sendet den Code aus dieser Datei, denn das Archiv wird nur mit einem gültigen Registrierungscode ausgeliefert (siehe „Paket-Downloads“):

```
printf 'X-CLM-Enrollment-Token: %s\nX-CLM-Tenant: %s\n' "$(cat clm-enrollment.token)" '<ORGANIZATION-ID>' | curl -fsSLO -H @- <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz.sha256
sha256sum -c clm-agent-linux-x64.tar.gz.sha256
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-linux-x64.tar.gz.sha256.p7s
```

```
curl -fsSLO <CONTROL-PLANE-ORIGIN>/downloads/verify-agent-package.sh
bash verify-agent-package.sh --publisher-sha256 <PUBLISHER-SHA256> clm-agent-linux-x64.tar.gz
tar -xzf clm-agent-linux-x64.tar.gz
cd clm-agent
```

`sha256sum` muss `clm-agent-linux-x64.tar.gz: OK` ausgeben; andernfalls nicht installieren. Die Prüfsumme erkennt einen beschädigten Download. `verify-agent-package.sh` muss `VERIFIED` ausgeben; `REFUSED: <Grund>` bedeutet: nicht installieren (siehe „Paketprüfung“). Wenn Ihr Betreiber einen Release-Signaturschlüssel veröffentlicht, prüfen Sie zusätzlich das signierte Manifest wie in Schritt 2 von „Fünf Schritte“ beschrieben. Meldet `curl` ein Zertifikatsproblem, lesen Sie zuerst „Private CA“ weiter unten; `The requested URL returned error: 401` beim ersten Befehl bedeutet, dass der Code fehlt, abgelaufen oder verbraucht ist oder die Organisations-ID falsch ist (siehe „Paket-Downloads“).

## 7.4 Schritt 4: Installationsprogramm ausführen

Im von `tar` angelegten Verzeichnis clm-agent (die Token-Datei bleibt im darüberliegenden Arbeitsverzeichnis, daher `../clm-enrollment.token`):

```
sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> bash install-linux.sh <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> ../clm-enrollment.token
```

Der Fingerabdruck nach `CLM_AGENT_PUBLISHER_SHA256=` ist erforderlich: Enthält der aus der Konsole kopierte Befehl ihn nicht, ergänzen Sie ihn wie gezeigt. Bevor der Host verändert wird, kopiert das Installationsprogramm das Paket in ein root-eigenes privates Staging-Verzeichnis unter /var/tmp, prüft diese Kopie (`SHA256SUMS`, dessen Signatur und den fixierten Herausgeber) und installiert genau die geprüfte Kopie. `CLM_AGENT_PUBLISHER_CA_FILE` verlangt zusätzlich die CA-Kette; `CLM_AGENT_VERIFIER=/pfad/verify-agent-package.sh` verwendet Ihren separat heruntergeladenen Prüfer. `bash install-linux.sh --verify-only` prüft das entpackte Paket ohne Installation. Ohne fixierten Fingerabdruck oder mit einem unsignierten oder veränderten Paket wird nichts installiert und der Registrierungscode nicht verbraucht. `CLM_AGENT_ALLOW_UNSIGNED=1` ist eine reine Entwicklungsoption, die eine Warnung ausgibt. `CLM_AGENT_ENDPOINT_VERIFIER=<Datei>` macht den Host zusätzlich zum unabhängigen Endpunkt-Prüfer eines Bereitstellungsziels (siehe „Unabhängige Verifizierung im privaten Netzwerk“).

Mit einer privaten Stammzertifizierungsstelle (Vertrauensschritt der Konsole) übergibt der von der Konsole angezeigte Befehl zusätzlich `CLM_AGENT_TRUST_ROOT=../certyx-agent-root.crt`: Das Installationsprogramm prüft, dass die Datei ein CA-Zertifikat ist, das noch mindestens einen Tag gültig ist, und fügt es vor seiner Verbindungsprüfung unter dem Namen `certyx-agent-root.crt` dem System-Zertifikatsspeicher der Distribution hinzu (Debian und Ubuntu: /usr/local/share/ca-certificates mit `update-ca-certificates`; RHEL-kompatibel: /etc/pki/ca-trust/source/anchors mit `update-ca-trust`; SUSE: /etc/pki/trust/anchors). Ein identischer Eintrag bleibt unverändert; schlägt die Verbindungsprüfung fehl, wird ein in diesem Lauf hinzugefügter Eintrag wieder entfernt, sodass der Zertifikatsspeicher des Hosts unverändert bleibt.

Das Installationsprogramm prüft jedes Argument, bevor es etwas ändert, und nennt ein falsches Argument zusammen mit dem nächsten Schritt. Dann führt es `Clm.Agent check` gegen die Adresse aus (DNS, TCP, TLS-Vertrauen mit Online-Sperrprüfung, Uhrzeit, Agent-Routen); ein fehlgeschlagener Schritt wird mit seiner Ursache ausgegeben, es wird nichts geändert und der Registrierungscode wird nicht verbraucht, sodass Sie das Problem beheben und denselben Befehl erneut ausführen können. Danach legt es das nicht privilegierte Konto clm-agent an, installiert das Release unter /opt/clm-agent/releases/<version>, wobei /opt/clm-agent/current darauf verweist, registriert den Agenten (der Agent erzeugt seinen RSA-3072-Schlüssel auf diesem Server; der Schlüssel verlässt ihn nie), löscht seine eigene Kopie des Tokens, hält die private Konfiguration, Identität und den Zustand unter /etc/clm-agent (`agent.json`, `identity.pfx`, `identity.password`, `state/`; nur für das Konto clm-agent zugänglich) und schreibt und startet die systemd-Unit `/etc/systemd/system/clm-discovery.service`, die `/opt/clm-agent/current/Clm.Agent scan /etc/clm-agent/agent.json` als clm-agent ausführt. Zum Schluss installiert es das Hilfsprogramm `/usr/local/bin/clm-agent` (das `uninstall-linux.sh` des Pakets bleibt für das Entfernen in /opt/clm-agent/current), wartet, bis CERTYX den ersten Kontakt des laufenden Dienstes erfasst, gibt `sudo clm-agent status` aus und endet mit „registered and running“. Löschen Sie danach clm-enrollment.token aus dem Arbeitsverzeichnis; der Code ist verbraucht. Weder Raw-Socket-Berechtigungen noch ein Root-Daemon sind erforderlich; Root-Rechte werden nur zum Anlegen von Konto und Dienst benötigt.

## 7.5 Schritt 5 (optional): signierte Updates

Nur wenn Ihr CERTYX-Betreiber einen Release-Signaturschlüssel veröffentlicht. Beziehen Sie release-signing.pub und seinen SHA-256-Fingerabdruck über einen vertrauenswürdigen Kanal vom Betreiber (nicht nur vom selben Server), kopieren Sie den Schlüssel neben das Installationsprogramm und geben Sie ihn als viertes Argument an:

```
sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> bash install-linux.sh <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> ../clm-enrollment.token ./release-signing.pub
```

Das Installationsprogramm gibt den Fingerabdruck des Schlüssels aus; vergleichen Sie ihn mit dem veröffentlichten. Der Update-Dienst lädt neue Archive mit dem Client-Zertifikat dieses Agents herunter, das das Installationsprogramm einrichtet (`AgentConfigurationFile` in update.json). Ohne das vierte Argument funktioniert der Agent, wird aber nicht automatisch aktualisiert; um signierte Updates später zu aktivieren, führen Sie `sudo bash upgrade-linux.sh ./release-signing.pub` aus einem neueren entpackten Paket aus (siehe „Direktes Upgrade“).

## 7.6 Schritt 6: prüfen, dass der Agent funktioniert

```
sudo clm-agent status
```

Das Installationsprogramm hat dies bereits einmal ausgegeben; führen Sie es jederzeit aus. Jede Zeile muss mit `[ OK ]` beginnen und die letzte Zeile `Result: connected` lauten, zum Beispiel:

```
[ OK ] Service: clm-discovery.service is active (running).
CERTYX agent 1.0.0: status of /etc/clm-agent/agent.json
[ OK ] Configuration: agent address https://agent.your-company.com, organization acme.
[ OK ] Identity: SHA-256 3BCC36CD…43A67FC0, valid until 2027-04-08; the private key stays on this host.
```

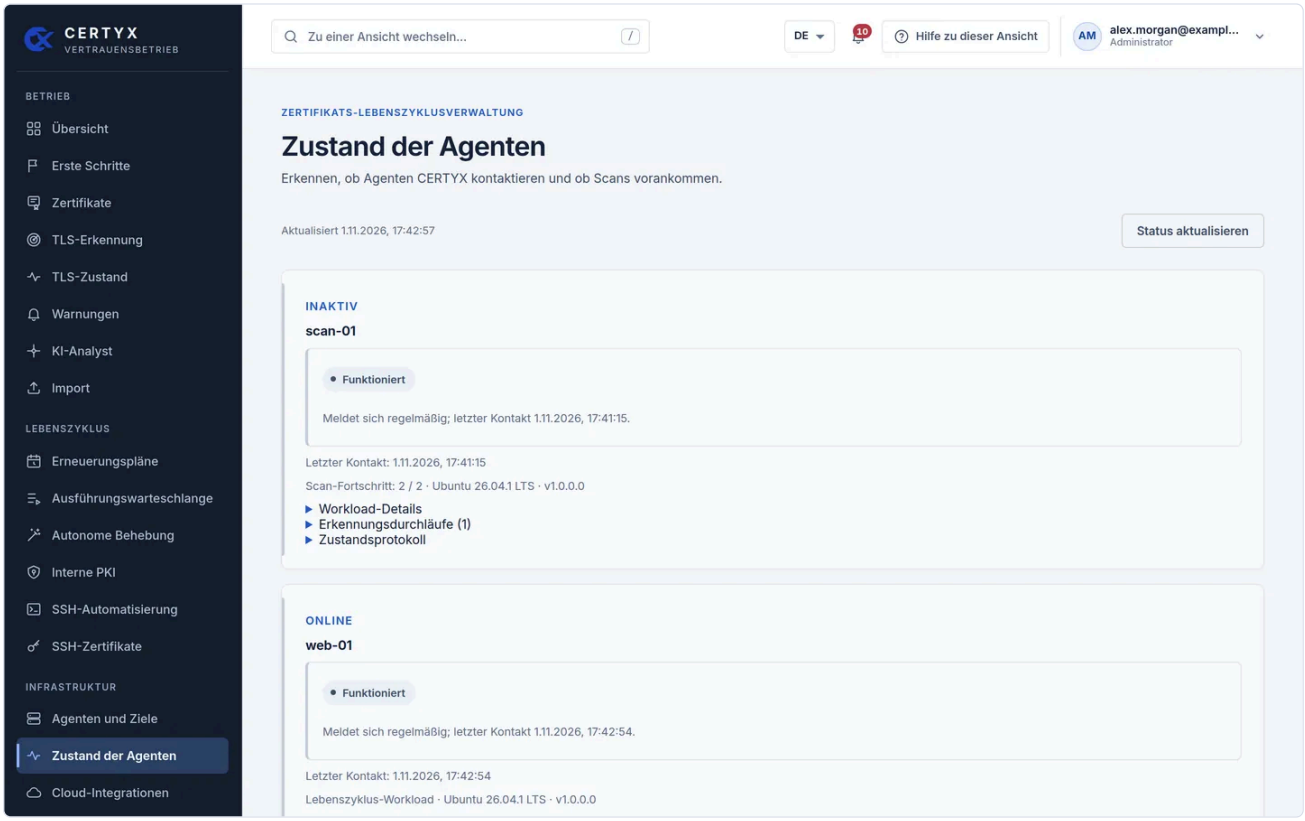
```
[ OK ] DNS: agent.your-company.com resolves to 10.20.0.15.
[ OK ] TCP: connected to 10.20.0.15:443 in 5 ms.
[ OK ] TLS: certificate for agent.your-company.com issued by Company Issuing CA is trusted, matches agent.your-company.com, is not revoked and is valid until 2027-11-11.
[ OK ] Clock: this host's clock matches the server (difference 0 s).
[ OK ] Agent address: agent.your-company.com serves agents.
[ OK ] Registration: CERTYX accepts this host as agent "berlin-dc1" in organization "acme"; first contact 2026-10-10 18:30:33 UTC, last contact 2026-10-10 18:31:03 UTC.
Result: connected. CERTYX accepts this agent; its contact times are shown above and in CERTYX -> Agents and destinations.
```

Der Befehl läuft unter dem Agent-Konto und prüft den Dienst, die Konfiguration, die lokale Identität (Fingerabdruck und Gültigkeit des Zertifikats), die Namensauflösung, die TCP-Verbindung, das TLS-Vertrauen einschließlich Sperrprüfung, die Uhrzeitabweichung und ob CERTYX die Identität akzeptiert, zusammen mit dem von CERTYX erfassten ersten und letzten Kontakt. Exit-Code 0 bedeutet funktionsfähig, 1 ein Problem (die erste `[FAIL]`-Zeile nennt es, siehe „Fehlerbehebung“). `sudo clm-agent check` führt nur die Dienst- und Verbindungsschritte aus; `sudo clm-agent log [Zeilen]` zeigt die letzten Protokollzeilen des Dienstes (Standard 50); `clm-agent version` gibt das installierte Release aus. In CERTYX zeigt die Konsole direkt nach „Registrierungscode erstellen“ „Agent … funktioniert: erster Kontakt empfangen am …“, sobald sich der Dienst meldet, und Registrierte Agents zeigt den ersten und letzten Kontakt („noch kein Kontakt“, solange der Dienst CERTYX nicht erreicht hat). Der Agent wartet dann auf sein Netzwerkprofil („No network profile is assigned“), bis Sie mit Schritt 4 von „Fünf Schritte“ fortfahren. Beenden Sie das Scannen mit `sudo systemctl stop clm-discovery.service` oder durch Deaktivieren des Profils in CERTYX; `sudo systemctl status clm-discovery.service` und `sudo journalctl -u clm-discovery.service` zeigen den Dienst selbst.

## 7.7 Private CA

Stammt das HTTPS-Zertifikat von CERTYX von der eigenen CA Ihrer Organisation, muss der Server dieser Stammzertifizierungsstelle vor der Verbindungsprüfung in Schritt 4 vertrauen. Veröffentlicht der Betreiber das Stammzertifikat mit `Clm:Ingress:AgentTrustRootFile`, wird nichts von Hand eingegeben: Der Vertrauensschritt der Konsole lädt es herunter und prüft seine SHA-256-Prüfsumme, und das Installationsprogramm fügt es mit `CLM_AGENT_TRUST_ROOT` dem System-Zertifikatsspeicher hinzu (Debian und Ubuntu, RHEL-kompatibel und SUSE) und entfernt es nach einer fehlgeschlagenen Prüfung wieder; `uninstall-linux.sh` entfernt es zusammen mit dem Agenten. Ohne veröffentlichtes Stammzertifikat erhalten Sie es von Ihrem PKI-Team und übergeben die Datei auf dieselbe Weise: `sudo CLM_AGENT_PUBLISHER_SHA256=<PUBLISHER-SHA256> CLM_AGENT_TRUST_ROOT=<ROOT-CA-FILE> bash install-linux.sh …` (für den Paket-Download `curl --cacert <ROOT-CA-FILE>`). Das Installationsprogramm lehnt eine Datei ab, die kein CA-Zertifikat ist oder innerhalb eines Tages abläuft, und bricht auf einer Distribution ohne `update-ca-certificates` oder `update-ca-trust` mit einer Meldung ab. Der Agent prüft außerdem online den Sperrstatus des CERTYX-Zertifikats; dieses Zertifikat muss daher einen CRL-Verteilungspunkt (oder OCSP-Responder) nennen, den der Server erreicht. Öffentliche CAs stellen beides bereit.

## 7.8 Diagnose in der Konsole




Zustand der Agenten: jeder Agent mit Zustand, letztem Kontakt, Workloads und der Ursache eines Problems (Beispieldaten)

Der Agentenzustand zeigt für jeden Agenten die genaue Ursache, die CERTYX aus Kontakt, Identität und Meldungen ableitet — meldet sich seit einem Zeitpunkt nicht, noch kein erster Kontakt, Identität abgelaufen oder läuft ab, Uhrzeitabweichung (gemessen am Zeitstempel der Netzwerkkarte), Prozessfehler mit dem vom Agenten gemeldeten festen Grund, ein Prozess, der sein Meldefenster verpasst hat, fehlende Konfiguration, reduzierte Scanrate — mit den Fakten und den Abhilfen der Reihe nach, jeweils mit Link zum Ort der Umsetzung. Jede Änderung der Ursache bleibt im Zustandsprotokoll des Agenten und im

Sicherheitsprotokoll ( `agent.health.changed` ). Ein Lebenszyklusagent meldet auch seine Ziele (ID, DNS-Namen, Konnektor, Installationsmodus, Remote-Host eines SSH-Relays, Bereitstellungsverzeichnis, Dateien, Prüf- und Reload-Befehle, Cloud-Ressource, Endpunkt; keine Geheimnisse, SSH-Identitäten oder known_hosts-Inhalte), die die Erneuerungsbereitschaft eines Zertifikats mit dem Ausführungsziel vergleicht: Die Ziel-ID muss einer Ziel-ID des Agenten mit denselben Namen, demselben Zweck, demselben Installationsmodus und demselben Remote-Host entsprechen, sonst lehnt der Agent den Auftrag ab.

## 7.9 Fehlerbehebung unter Linux

Das Installationsprogramm, `Clm.Agent check <Adresse>` und `sudo clm-agent status` geben eine Zeile pro Schritt aus; beheben Sie die Ursache der ersten `[FAIL]` -Zeile und führen Sie den Befehl erneut aus. Die Meldungen (auf Englisch ausgegeben):

- `[FAIL] DNS: <Name> cannot be resolved from this host (HostNotFound)` : Der Name ist falsch oder der DNS-Server dieses Hosts kennt ihn nicht. Prüfen Sie die Adresse aus der Konsole und `/etc/resolv.conf` ; für einen Test tragen Sie den Namen in `/etc/hosts` ein.
- `[FAIL] TCP: cannot connect to <Host> port <Port> (… ConnectionRefused)` : Dort lauscht nichts. Prüfen Sie den Port in der Agent-Adresse; auf dem lokalen Testserver starten Sie CERTYX mit `CLM_AGENT_HOST` und erlauben eingehendes TCP 5443.
- `[FAIL] TCP: … (… TimedOut)` : Eine Firewall verwirft die Verbindung oder es gibt keine Route. Erlauben Sie ausgehendes TCP zum Port (und eingehend auf dem Server).
- `[FAIL] TLS: no TLS handshake with …` : Der Port spricht kein HTTPS, zum Beispiel der HTTP-Konsolenport 5180. Verwenden Sie die HTTPS-Agent-Adresse.
- `[FAIL] TLS: this host does not trust the issuer of the server certificate (…)` : Fügen Sie das private oder lokale Stammzertifikat hinzu (Vertrauensschritt der Konsole oder „Private CA“) und führen Sie den Befehl erneut aus.
- `[FAIL] TLS: revocation of the server certificate cannot be checked: this host cannot download the revocation list http://…` : Erlauben Sie ausgehendes HTTP zur genannten Adresse (auf dem lokalen Testserver: eingehendes TCP 5181 und ein laufendes `./start.sh` ). `… the certificate names no CRL distribution point` bedeutet, dass der Betreiber ein Zertifikat einer CA verwenden muss, die eine CRL veröffentlicht.
- `[FAIL] TLS: a certificate of the chain is outside its validity period (…; this host's time is …)` : Prüfen Sie die Uhr dieses Hosts; stimmt sie, muss der Betreiber das Serverzertifikat erneuern.
- `[FAIL] TLS: the server certificate does not name <Host>` : Verwenden Sie genau die Adresse aus der Konsole; das Zertifikat muss diesen Namen oder diese IP-Adresse enthalten.
- `[FAIL] Clock: this host's clock differs from the server by <n> minutes` : Registrierung und Authentifizierung schlagen bei mehr als einigen Minuten Abweichung fehl. Führen Sie `sudo timedatectl set-ntp true` aus (oder konfigurieren Sie NTP) und versuchen Sie es erneut.
- `[FAIL] Agent address: <Adresse> does not serve agents (HTTP 404)` : Sie haben die Konsolenadresse verwendet, obwohl die Installation eine eigene Agent-Adresse hat; verwenden Sie die „Adresse für Agents“.
- `[FAIL] Registration: CERTYX does not accept this agent identity (HTTP 401)` : Der Agent wurde widerrufen, sein Zertifikat ist abgelaufen oder er gehört zu einer anderen Organisation oder Installation. Ersetzen Sie ihn wie bei „already has an enrolled CERTYX agent“ beschrieben.
- `[FAIL] Service: clm-discovery.service is inactive` : Starten Sie ihn mit `sudo systemctl start clm-discovery.service` und lesen Sie `sudo clm-agent log` .
- `[FAIL] Configuration: cannot read the agent configuration` : Führen Sie den Befehl mit sudo aus; fehlt die Datei, ist der Agent nicht installiert.

Meldungen beim Download:

- `curl: (22) The requested URL returned error: 401` für `clm-agent-linux-x64.tar.gz` : Der Archiv-Download enthielt keinen gültigen Registrierungscode. Führen Sie den Befehl im Arbeitsverzeichnis mit clm-enrollment.token aus; liegt die Datei dort, ist der Code abgelaufen oder verbraucht oder die Organisations-ID falsch. Erstellen Sie einen neuen Code, kopieren Sie die neue Datei und führen Sie die Befehle erneut aus (siehe „Paket-Downloads“).

Meldungen des Installationsprogramms:

- `Package verification failed; nothing was installed.` : Das Paket ist unsigniert, wurde nach dem Signieren verändert oder von einem anderen Herausgeber signiert als dem in `CLM_AGENT_PUBLISHER_SHA256` genannten. Prüfen Sie den Fingerabdruck (unabhängig vom Download von Ihrem CLM-Anbieter bezogen), entpacken Sie das Paket erneut aus dem geprüften Archiv und fügen Sie keine Dateien hinzu oder ändern sie.
- `Package verifier missing; nothing was installed.` : `verify-agent-package.sh` fehlt im entpackten Paket oder `CLM_AGENT_VERIFIER` nennt keine reguläre Datei. Entpacken Sie das geprüfte Archiv erneut oder lassen Sie `CLM_AGENT_VERIFIER` auf den separat heruntergeladenen Prüfer zeigen.
- `Enrollment token file not found: <Pfad>` : Das dritte Argument zeigt nicht auf die Token-Datei auf diesem Server. Laden Sie sie herunter (Schritt 1), kopieren Sie sie in das Arbeitsverzeichnis (Schritt 2), führen Sie das Installationsprogramm im darunterliegenden Verzeichnis clm-agent aus und übergeben Sie `../clm-enrollment.token` . Die Meldung zeigt auch das aktuelle Verzeichnis.
- `Argument 1 is still a placeholder, the example address from the guide` oder `not a valid control-plane origin` : Verwenden Sie die „Adresse für Agents“ aus der Konsole: `https://` , den Hostnamen und optional einen Port, ohne Pfad.
- `The enrollment code was rejected` (HTTP 400, 401, 403 oder 409): Der Code ist abgelaufen (er gilt für die in `Clm:Agents:Enrollment:TicketLifetime` festgelegte Dauer, standardmäßig zehn Minuten; die Konsole zeigt die Frist), wurde bereits verwendet oder die Organisations-ID passt nicht. Erstellen Sie einen neuen Code für dieselbe Agenten-ID, kopieren Sie die neue Datei und führen Sie denselben Befehl erneut aus; der lokale Schlüssel des fehlgeschlagenen Versuchs wird wiederverwendet.
- `does not serve agents` (HTTP 404): Sie haben die Konsolenadresse verwendet, obwohl Ihre Installation eine eigene Agentenadresse hat. Verwenden Sie die „Adresse für Agents“ aus der Konsole.
- `cannot be reached` oder `cannot be resolved` : Prüfen Sie Adresse, Port, DNS und die Firewallregel für ausgehendes HTTPS. `curl -s -o /dev/null -w '%{http_code}\n' <CONTROL-PLANE-ORIGIN>/health/live` muss 200 ausgeben.
- `does not trust the HTTPS certificate` : siehe „Private CA“.
- `already has an enrolled CERTYX agent` : Dieser Server ist bereits registriert, und es wurde nichts geändert; prüfen Sie ihn mit `sudo clm-agent status` . Zum Ersetzen entfernen Sie ihn mit `sudo bash /opt/clm-agent/current/uninstall-linux.sh` (siehe „Entfernen unter Linux“), widerrufen Sie den Agenten in CERTYX, erstellen Sie dann einen Code für eine neue Agenten-ID (eine Agenten-ID kann nur einmal registriert werden) und installieren Sie erneut.

- `clm-discovery.service is not running`: Das Installationsprogramm gibt die letzten Protokollzeilen aus. Mehr lesen Sie mit `sudo journalctl -u clm-discovery.service -n 50 --no-pager`; Fehler werden nach Kategorie (Ausnahmetyp, HTTP-Status) protokolliert, nie mit Tokens oder Schlüsseln.
- `This host does not run systemd`: Verwenden Sie „Manuelle Ausführung / Ausführung im Container“.

### 7.10 Entfernen unter Linux

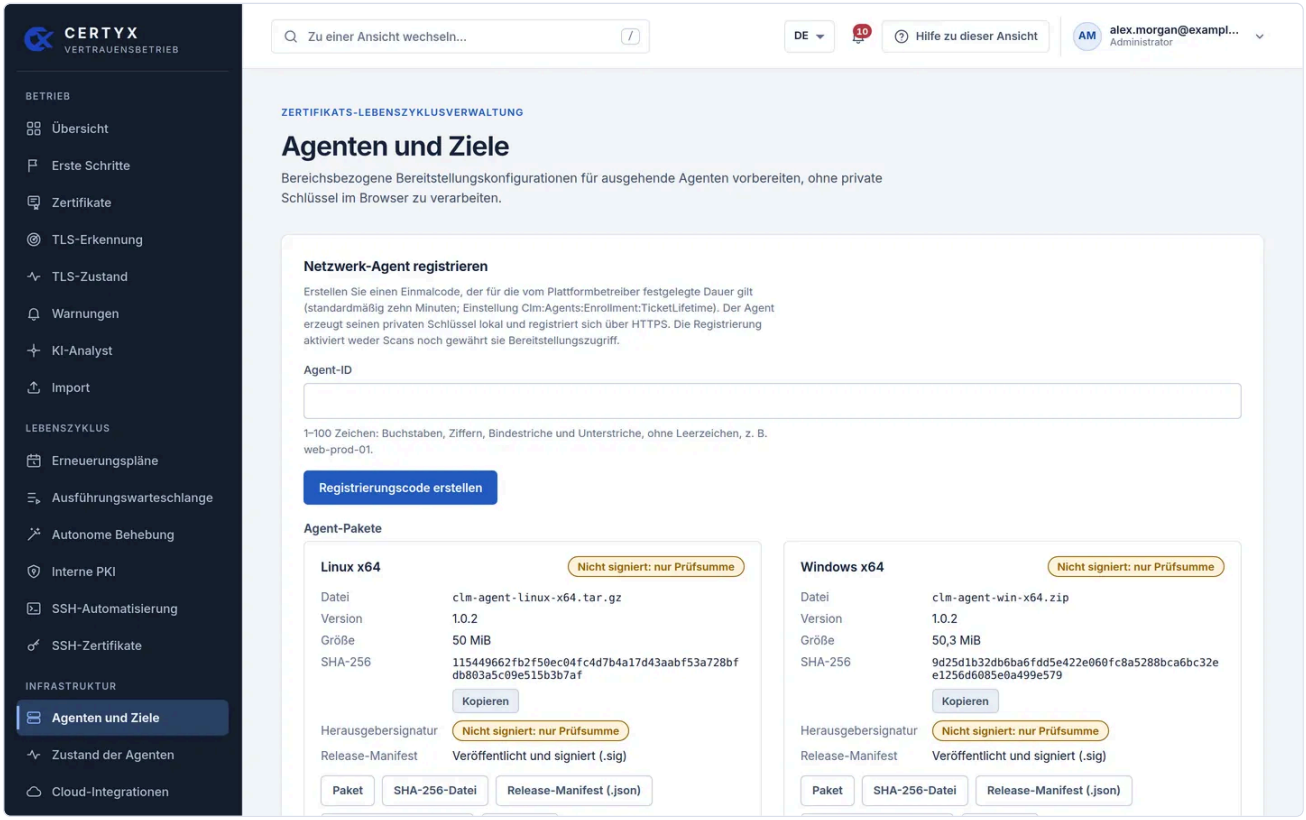
```
sudo bash /opt/clm-agent/current/uninstall-linux.sh
```

Das Skript arbeitet aus einer temporären Kopie weiter (es löscht sein eigenes Verzeichnis), stoppt, deaktiviert und löscht jede Unit, die die CERTYX-Skripte installiert haben (`clm-discovery.service`, `clm-ssh.service` mit seinem Drop-in für die Host-Bereitstellung, `clm-ssh-renewal.service` und `.timer`, `clm-endpoint-verifier.service`, `clm-agent-update.service` und `.timer`), entfernt den CERTYX-Block, den `trust-user-ca` in /etc/ssh/sshd_config eingefügt hat (geprüft mit `sshd -t`; eine Sicherung wird aufbewahrt und wiederhergestellt, wenn sshd das Ergebnis ablehnt; `--keep-ssh-trust` lässt sshd unverändert), die sudoers-Regel der SSH-Host-Bereitstellung und den vom Installationsprogramm hinzugefügten Vertrauensanker `certyx-agent-root.crt` (`--keep-trust-root` behält ihn) und löscht /opt/clm-agent, /etc/clm-agent, /var/lib/clm-agent, /etc/clm-agent-update, /etc/clm-ssh, /etc/clm-verifier, /usr/local/bin/clm-agent und das Konto clm-agent. Die privaten Schlüssel werden mit diesen Verzeichnissen gelöscht. Ein erneuter Lauf auf einem bereinigten Host ändert nichts. Widerrufen Sie danach den Agenten in CERTYX → Agenten und Ziele (Registrierte Agenten → Agenten widerrufen): Erst dann akzeptiert CERTYX seine Identität nicht mehr.

## 8 Windows mit Aufgabenplanung

Wie unter Linux zeigt die Konsole nach „Registrierungscode erstellen“ diese Befehle mit Ihren echten Werten an. Die Platzhalter `<CONTROL-PLANE-ORIGIN>`, `<ORGANIZATION-ID>` und `<PUBLISHER-SHA256>` bedeuten dasselbe wie im Linux-Abschnitt; ersetzen Sie sie einschließlich der spitzen Klammern.

### 8.1 Schritt 1: Registrierungscode erstellen




Agenten und Ziele: Registrierung, registrierte Agenten, Netzwerkprofile und Zielentwürfe (Beispieldaten)

Wie in Linux-Schritt 1: Erstellen Sie den Code in CERTYX und laden Sie clm-enrollment.token herunter. Er gilt nur einmal und für die konfigurierte Dauer (`Clm:Agents:Enrollment:TicketLifetime`, standardmäßig zehn Minuten; die Konsole zeigt die Frist); führen Sie die Schritte 2–4 in dieser Zeit aus.

### 8.2 Schritt 2: Token-Datei in einen Arbeitsordner kopieren

Legen Sie auf dem Windows-Server einen leeren Arbeitsordner an und kopieren Sie clm-enrollment.token hinein. Behalten Sie den Dateinamen bei. Führen Sie die Befehle der Schritte 3 und 4 dort in einer Administrator-PowerShell aus.

## 8.3 Schritt 3: Paket herunterladen, prüfen und entpacken

Zeigt die Konsole den Vertrauensschritt für ihre eigene Zertifizierungsstelle, führen Sie ihn zuerst in einer Administrator-PowerShell aus; der zweite Befehl muss `True` ausgeben:

```
curl.exe -fsSLk -o certyx-agent-root.crt <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-trust-root.crt
(Get-FileHash certyx-agent-root.crt -Algorithm SHA256).Hash -eq '<SHA256-FROM-CONSOLE>'
Import-Certificate -FilePath .\certyx-agent-root.crt -CertStoreLocation Cert:\LocalMachine\Root
```

Danach in einer PowerShell mit Administratorrechten auf dem Windows-Server (x64) im Arbeitsordner mit clm-enrollment.token. Der erste Befehl liest den Code aus dieser Datei in die Header `$clmDownload`, die der Archiv-Download sendet (siehe „Paket-Downloads“):

```
$clmDownload = @{ 'X-CLM-Enrollment-Token' = (Get-Content .\clm-enrollment.token -Raw).Trim(); 'X-CLM-Tenant' = '<ORGANIZATION-ID>' }
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip -Headers $clmDownload -OutFile clm-agent-win-x64.zip -UseBasicParsing
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip.sha256 -OutFile clm-agent-win-x64.zip.sha256 -UseBasicParsing
(Get-FileHash clm-agent-win-x64.zip -Algorithm SHA256).Hash -eq (Get-Content clm-agent-win-x64.zip.sha256).Split(' ')[0]
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/clm-agent-win-x64.zip.sha256.p7s -OutFile clm-agent-win-x64.zip.sha256.p7s -UseBasicParsing
Invoke-WebRequest <CONTROL-PLANE-ORIGIN>/downloads/verify-agent-package.ps1 -OutFile verify-agent-package.ps1 -UseBasicParsing
Get-AuthenticodeSignature .\verify-agent-package.ps1
.\verify-agent-package.ps1 -PublisherSha256 <PUBLISHER-SHA256> -Archive .\clm-agent-win-x64.zip
Expand-Archive clm-agent-win-x64.zip -DestinationPath clm-agent
cd clm-agent
```

Der Vergleich mit `Get-FileHash` muss `True` und `verify-agent-package.ps1` muss `VERIFIED` ausgeben; andernfalls nicht installieren (siehe „Paketprüfung“). `(401) Unauthorized` beim Archiv-Download bedeutet, dass der Code fehlt, abgelaufen oder verbraucht ist oder die Organisations-ID falsch ist (siehe „Paket-Downloads“). Das signierte Manifest können Sie zusätzlich mit PowerShell 7 oder OpenSSL prüfen, wie unter „Signierte Updates“ beschrieben.

## 8.4 Schritt 4: Installationsprogramm ausführen und Aufgabe prüfen

Im von `Expand-Archive` angelegten Ordner clm-agent (die Token-Datei bleibt im darüberliegenden Arbeitsordner, daher `..\clm-enrollment.token`):

```
powershell -NoProfile -ExecutionPolicy Bypass -File .\install-windows.ps1 -ControlPlane <CONTROL-PLANE-ORIGIN> -Tenant <ORGANIZATION-ID> -TokenFile ..\clm-enrollment.token -PublisherSha256 <PUBLISHER-SHA256>
Get-ScheduledTask -TaskName 'CLM Discovery Agent' | Get-ScheduledTaskInfo
```

`-ExecutionPolicy Bypass` gilt nur für diesen einen Installationsprozess. Das Installationsprogramm prüft zuerst die Argumente und nennt ein falsches Argument mit dem nächsten Schritt (dieselben Meldungen wie unter Linux), dann führt es `Clm.Agent.exe check` gegen die Adresse aus, ohne den Code zu verbrauchen. Am Ende wartet es auf den ersten Kontakt und gibt die Statusprüfung aus. Führen Sie sie jederzeit in einer Administrator-PowerShell erneut aus; jede Zeile muss mit `[ OK ]` beginnen und die letzte `Result: connected` lauten:

```
& "$env:ProgramData\CLM-Agent\Clm.Agent.exe" status "$env:ProgramData\CLM-Agent\agent.json"
```

Das Installationsprogramm schränkt die Berechtigungen für ProgramData\CLM-Agent ein, macht Administratoren zum Besitzer, kopiert das Token für die Registrierung dorthin und löscht diese Kopie anschließend, und registriert die geplante Aufgabe `CLM Discovery Agent`, die beim Systemstart als LOCAL SERVICE startet, `C:\ProgramData\CLM-Agent\Clm.Agent.exe scan C:\ProgramData\CLM-Agent\agent.json` ohne Zeitlimit ausführt und nach einem Fehler bis zu dreimal im Abstand von einer Minute neu gestartet wird. Agent und Konfiguration (`agent.json`) liegen in `C:\ProgramData\CLM-Agent`, die Identität (`identity.pfx`, `identity.password`) in `C:\ProgramData\CLM-Agent\identity` und der Zustand in `state\`; nur diese beiden Unterverzeichnisse sind für LOCAL SERVICE beschreibbar: Die Identitätsrotation legt die nächste Identität (`identity.pfx.pending`) im Identitätsverzeichnis ab und benennt sie an ihren Platz um; sonst kann das Aufgabenkonto im Programmverzeichnis nichts ändern. Das Installationsprogramm weist ein nicht leeres ProgramData\CLM-Agent ab, kopiert das Paket in das geschützte Verzeichnis und prüft diese geschützte Kopie (signiertes Manifest, fixierter Herausgeber und die Authenticode-Signaturen) vor der Registrierung; ein abgewiesenes Paket wird wieder entfernt und nichts registriert. `-PublisherSha256` ist erforderlich; `-PublisherCaFile` verlangt zusätzlich die CA-Kette, `-VerifierPath` verwendet einen separat heruntergeladenen Prüfer und `-VerifyOnly` prüft nur das entpackte Paket. `-AllowUnsignedDevelopment` ist eine reine Entwicklungsoption, die eine Warnung ausgibt. Es wird kein Windows-Dienst angelegt. Löschen Sie danach clm-enrollment.token aus dem Arbeitsordner. Neuere Versionen werden mit `upgrade-windows.ps1` direkt installiert (siehe „Direktes Upgrade“).

**Automatische signierte Updates (optional):** Veröffentlicht Ihr Betreiber einen Release-Signaturschlüssel, ergänzen Sie den Installationsbefehl um `-ReleaseKeyFile .\release-signing.pub` (zusammen mit seinem Fingerabdruck über einen vertrauenswürdigen Kanal bezogen). Nach der Registrierung führt das Installationsprogramm das installierte `upgrade-windows.ps1 -EnableUpdates` aus: Es legt den Schlüssel und den Herausgeber-Fingerabdruck in `C:\ProgramData\CLM-Agent-Update` fest (nur Administratoren und LocalSystem) und registriert die geplante Aufgabe `CERTYX Agent Update`. Sie führt täglich um 03:00 Uhr mit einer zufälligen Verzögerung von bis zu sechs Stunden als LocalSystem `upgrade-windows.ps1 -Scheduled` aus, also dasselbe signierte `-Download`-Upgrade wie unter „Direktes Upgrade“ (einschließlich Rollback); ein nicht neueres Release endet mit Exit-Code 0, die Ausgabe wird an `C:\ProgramData\CLM-Agent-Update\update.log` angehängt. Prüfen Sie sie mit `Get-ScheduledTask -TaskName 'CERTYX Agent Update' | Get-ScheduledTaskInfo`. Eine Installation ohne Schlüssel erhält die Aufgabe mit ihrem ersten signierten Upgrade (`-ReleaseKeyFile`) oder mit `upgrade-windows.ps1 -EnableUpdates -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub`.

Die Aufgabe hat kein Konsolenfenster: Der Agent schreibt seine Ausgabe nach `C:\ProgramData\CLM-Agent\state\discovery.log` (Zeilen mit UTC-Zeitstempel, höchstens 5 MB; der vorherige Teil bleibt als `discovery.log.1` erhalten). Lesen Sie es in einer Administrator-PowerShell mit `Get-Content -Tail 50 C:\ProgramData\CLM-Agent\state\discovery.log`, dem Windows-Gegenstück zu `sudo clm-agent log`. Die Aufgabe des SSH-Workloads schreibt `C:\ProgramData\CLM-SSH\state\ssh-workload.log`; dessen letzte Zeilen zeigt `configure-ssh-windows.ps1 status`. Das Protokoll enthält nur die sicheren Zusammenfassungen des Agenten, niemals Tokens, Passwörter oder Schlüssel.

Verwendet CERTYX eine private CA ohne veröffentlichtes Stammzertifikat, importieren Sie zuerst das Stammzertifikat Ihres PKI-Teams: `Import-Certificate -FilePath <ROOT-CA-FILE> -CertStoreLocation Cert:\LocalMachine\Root`, wobei `<ROOT-CA-FILE>` der Pfad des Stammzertifikats ist, das Ihr PKI-Team Ihnen gibt. Zeigt die Konsole den Vertrauensschritt, verwenden Sie dessen Befehle. Die Anforderung an den CRL-Verteilungspunkt aus „Private CA“ gilt ebenfalls.

Fehlerbehebung: Meldungen und Abhilfen entsprechen denen für Linux. Um eine Installation zu ersetzen, entfernen Sie sie mit `uninstall-windows.ps1` (siehe „Entfernen“), widerrufen den Agenten in CERTYX und installieren erneut mit einem Code für eine neue Agenten-ID.

### 8.5 Entfernen

In einer Administrator-PowerShell:

```
powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\uninstall-windows.ps1"
```

Das Skript stoppt und entfernt die geplanten Aufgaben `CLM Discovery Agent`, `CERTYX SSH Workload`, `CERTYX SSH Renewal`, `CERTYX Endpoint Verifier` und `CERTYX Agent Update`, entfernt den CERTYX-Block, den `trust-user-ca` in `C:\ProgramData\ssh\sshd_config` eingefügt hat, zusammen mit `certyx_user_ca.pub` (mit `sshd -t` geprüft; eine Sicherung bleibt erhalten und wird wiederhergestellt, wenn sshd das Ergebnis nicht akzeptiert; `-KeepSshTrust` lässt sshd unverändert) und löscht `C:\ProgramData\CLM-Agent` (Programm, Konfiguration, Identität, Endpunkt-Prüfer, Zustand und Protokolle), `C:\ProgramData\CLM-SSH` (SSH-CA, Schlüssel und Workload-Zustand) und `C:\ProgramData\CLM-Agent-Update` (der festgelegte Release-Schlüssel, der Herausgeber-Fingerabdruck und das Update-Protokoll). Die privaten Schlüssel werden mit diesen Ordnern gelöscht. Widerrufen Sie den Agenten anschließend in CERTYX → Agenten und Ziele (Registrierte Agents → Agent widerrufen): Erst dann akzeptiert CERTYX seine Identität nicht mehr. Eine Agenten-ID kann nur einmal registriert werden; eine neue Installation braucht einen Code für eine neue Agenten-ID.

## 9 Manuelle Ausführung / Ausführung im Container

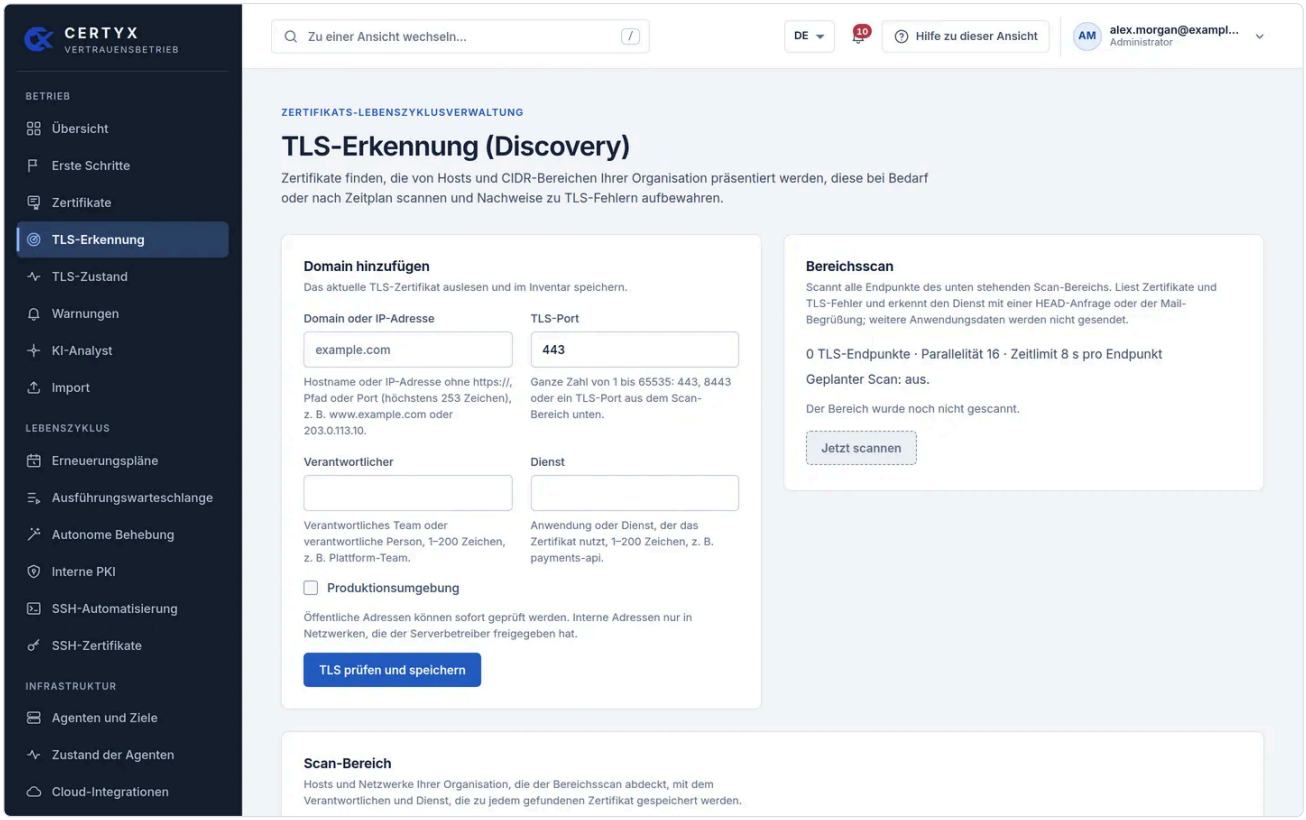
Dieselbe ausführbare Datei kann ohne Installationsprogramm des Betriebssystems laufen:

```
Clm.Agent enroll <CONTROL-PLANE-ORIGIN> <ORGANIZATION-ID> <ENROLLMENT-TOKEN-FILE> <CONFIG-FILE>
Clm.Agent scan <CONFIG-FILE>
Clm.Agent scan <CONFIG-FILE> --once
```

`<CONTROL-PLANE-ORIGIN>` und `<ORGANIZATION-ID>` sind die Werte, die die Konsole nach „Registrierungscode erstellen“ anzeigt; `<ENROLLMENT-TOKEN-FILE>` ist die heruntergeladene clm-enrollment.token (nur für den Besitzer lesbar, Modus 600); `<CONFIG-FILE>` ist der Pfad, unter den die Registrierung die Agentenkonfiguration schreibt, zum Beispiel /var/lib/clm-agent/agent.json; in sein Verzeichnis kommen auch die Identitätsdateien.

Binden Sie Konfigurations- und Identitätsdateien privat ein, speichern Sie das Statusverzeichnis dauerhaft und stellen Sie dieselbe geroutete Netzwerkerreichbarkeit wie bei einem Host-Agenten bereit. Container-Bridge-Netzwerke sind nicht automatisch das Kundennetzwerk: Die Netzwerkkarte eines Containers zeigt sein Bridge-Netz; genehmigen Sie die erreichbaren CIDRs daher ausdrücklich und setzen Sie `RestrictToAllowedCidrs` mit `AllowedCidrs`, wenn der Container-Host eine eigene Grenze durchsetzen muss. --once verarbeitet einen begrenzten Stapel (höchstens die Versuche pro Zyklus des Profils), nicht das gesamte Netzwerk.

## 10 Abdeckung und Fortschritt




TLS-Erkennung (Discovery): Umfang, Fortschritt und die im privaten Netzwerk gefundenen Endpunkte und Zertifikate (Beispieldaten)

Jeder Durchlauf hat bis zu drei Phasen, immer in dieser dokumentierten Reihenfolge:

1. **Erreichbarkeit (liveness):** jede Adresse der genehmigten CIDRs (ohne Ausnahmen und, bis /30, ohne Netz- und Broadcast-Adresse) in aufsteigender Reihenfolge. Der Agent versucht die Erreichbarkeitsports nacheinander; eine Adresse ist aktiv, wenn ein Port die Verbindung annimmt oder aktiv ablehnt (TCP-Reset). Adressen aus der ARP-Nachbartabelle des Hosts gelten ohne Verbindung als aktiv.
2. **Gängige Ports (common):** jeder aktive Host in aufsteigender Reihenfolge, seine gängigen Ports in der Reihenfolge des Profils, mit TCP-Verbindung und direktem TLS-Handshake, gefolgt von der unter „Für das Sicherheitsteam des Kunden“ beschriebenen Diensterkennung.
3. **Vollständig (full,** nur im Modus `full`): jeder verbleibende Port 1–65535 in aufsteigender Reihenfolge, jeder Port auf allen aktiven Hosts vor dem nächsten Port, sodass aufeinanderfolgende Verbindungen verschiedene Hosts treffen.

Es gibt keine Zufallsreihenfolge und keinen Jitter. Der Agent öffnet nie mehr als die im Profil festgelegten Verbindungen pro Host (1–4) zu einem Host und baut Verbindungen mit einer gleichmäßigen Rate von höchstens `ratePerSecond` (1–2000) auf; das lokale `ProbeSpacingMilliseconds` kann ihn nur verlangsamen. Außerhalb der Scanfenster (wöchentliche Fenster in der Zeitzone des Profils; ein Ende vor dem Beginn reicht in den nächsten Tag, `24:00` endet um Mitternacht, Fenster folgen bei Sommer-/Winterzeitwechseln der lokalen Uhrzeit) baut der Agent keine Verbindungen auf und meldet den Zustand `waiting-window` mit dem Beginn des nächsten Fensters. Wenn in einem Segment der gängigen oder vollständigen Phase mehr als `BackOffFailurePercent` (Standard 50 %) der Verbindungsversuche in eine Zeitüberschreitung laufen, wird die Rate des nächsten Segments halbiert (nie unter `BackOffMinimumPercent`, Standard 10 %, der genehmigten Rate) und nach jedem ruhigeren Segment um `BackOffRestorePercent` (Standard 25 %) der genehmigten Rate wieder angehoben; die Telemetrie zeigt `backOff`. Die Telemetrie meldet außerdem die Phase, die Position in der Phase, aktive Hosts und `baselineCompletedAt`, den Zeitpunkt, zu dem der erste vollständige Durchlauf des aktuellen Bereichs endete.

Der Fortschritt wird nach jedem Segment (`SegmentSize`, Standard 200 Arbeitsschritte) im nur für den Eigentümer lesbaren Prüfpunkt gespeichert: Phase, Position und aktive Hosts. Der Prüfpunkt ist nur an einen Hash von Bereich und Ports gebunden; Änderungen an Rate, Verbindungen pro Host, Fenstern oder Stapelwerten setzen den Durchlauf fort, ein geänderter Bereich, Modus oder eine geänderte Portliste beginnt ihn neu. Das Profil wird zwischen den Segmenten erneut geprüft; ein Stapel endet nach den Versuchen pro Zyklus des Profils, und der nächste Stapel beginnt nach dem Intervall des Profils. Eine von CLM mit HTTP 400 abgelehnte Beobachtung (zum Beispiel, weil der Bereich inzwischen verkleinert wurde) wird verworfen, in der Telemetrie gemeldet, und der Scan läuft weiter. Ein fehlerhaftes Profil wird bei der Validierung abgelehnt und als ungültig gemeldet; der Agent läuft weiter und versucht es nach `RetrySeconds` erneut.

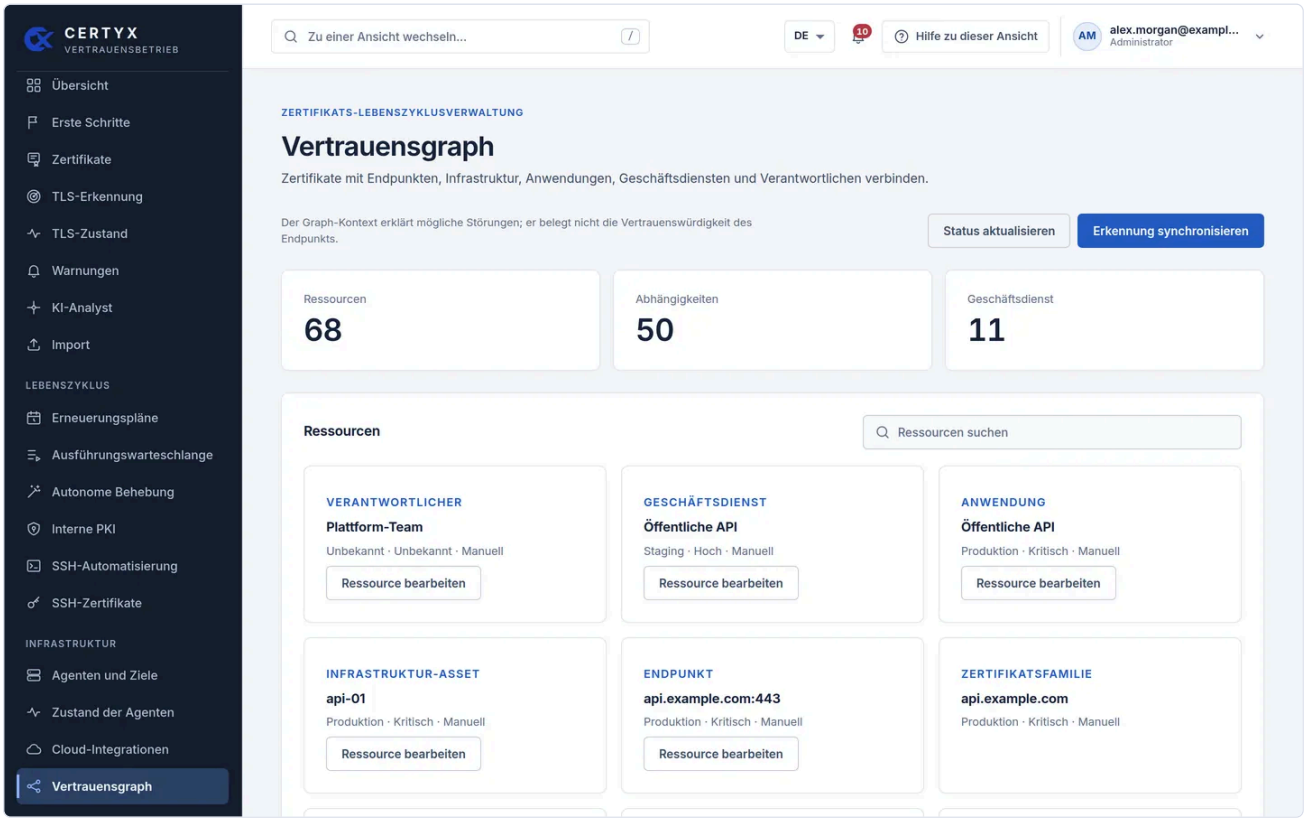
Realistische Laufzeiten ergeben sich aus den Arbeitsschritten geteilt durch die wirksame Rate. Diese ist der kleinere Wert aus `ratePerSecond` und `Concurrency` geteilt durch die Verbindungszeitüberschreitung (wenn die meisten Versuche ins Leere laufen). Stapel (`attemptsPerCycle`) begrenzen nur die Arbeit zwischen Profilaktualisierungen und Prüfpunkten und folgen unmittelbar aufeinander; `intervalSeconds` ist die Pause nach einem vollständigen Durchlauf:

- Erreichbarkeit eines /24 mit 4 Erreichbarkeitsports: höchstens etwa 1.000 Versuche, etwa 10 Sekunden bei 100 pro Sekunde. Ein dünn belegtes /16: etwa 262.000 Versuche, etwa 45 Minuten bei 100 pro Sekunde oder 9 Minuten bei 500 pro Sekunde. Nicht vorhandene Adressen laufen in Zeitüberschreitungen, daher erlauben 32 parallele Verbindungen mit der Verbindungszeitüberschreitung von 500 ms nur etwa 64 Versuche pro Sekunde (etwa 70 Minuten).
- Gängige Ports: aktive Hosts × gängige Ports, zum Beispiel 300 Hosts × 20 Ports = 6.000 Versuche, etwa eine Minute bei 100 pro Sekunde.
- Modus `full`: aktive Hosts × 65.535 Ports, zum Beispiel 50 Hosts = etwa 3,3 Millionen Versuche, etwa 4,6 Stunden bei 200 pro Sekunde oder 55 Minuten bei 1.000 pro Sekunde. Firewalls, die Pakete verwerfen, machen die meisten Versuche zu Zeitüberschreitungen mit Drosselung; das kann die Dauer auf einen Tag oder mehr verlängern. Mit 10.000 Versuchen pro Zyklus, 200 pro Sekunde und einem Intervall von 300 Sekunden liegt der Durchschnitt bei etwa

29 Versuchen pro Sekunde, derselbe Durchlauf dauert dann etwa 32 Stunden; verkürzen Sie das Intervall, wenn die Fenster und Ihr Sicherheitsteam es erlauben.

Dass kein Zertifikat erkannt wurde, beweist nicht, dass keines vorhanden ist. Die Erkennung erfasst derzeit direktes TLS über IPv4-TCP. UDP/DTLS/QUIC, SMTP/LDAP STARTTLS, IPv6-Adressscans, die Aufzählung virtueller Hosts per SNI, Zertifikatsspeicher im Dateisystem und Cloud-Connectoren sind in diesem Scanner nicht enthalten.

## 11 Netzwerkkarte




Vertrauensgraph: Hosts, Dienste und Zertifikate mit ihren Abhängigkeiten (Beispieldaten)

Beim Start und danach alle `NetworkMapIntervalMinutes` (Standard 60) sendet der Agent CLM die privaten IPv4-Netze seines Hosts: direkt angeschlossene Netze aktiver Schnittstellen und Ziele der Routingtabelle (Linux `/proc/net/route`, Windows-Routingtabelle der IP-Hilfsfunktionen), dazu private Adressen der ARP-Nachbartabelle (Linux `/proc/net/arp`, Windows-Nachbartabelle der IP-Hilfsfunktionen) als Hinweise auf aktive Hosts. Gemeldet werden nur RFC-1918-Bereiche und 100.64.0.0/10; Standardrouten, Loopback, Link-Local, Multicast, öffentliche Ziele und alles, was breiter als /8 ist, werden verworfen, und es werden höchstens 1.024 Netze und 4.096 Nachbarn gesendet. Das Lesen dieser Tabellen sendet keine Pakete. Die Karte ist nur ein Vorschlag für den Administrator: Gescannt wird erst, wenn ein Bereich in einem Profil genehmigt ist. Lehnt CLM die Karte ab (zum Beispiel ein älterer Server ohne diesen Endpunkt), protokolliert der Agent den HTTP-Status und versucht es beim nächsten Intervall erneut; der Scan ist davon nicht betroffen. IPv6-Nachbartabellen werden nicht gelesen; auf anderen Betriebssystemen werden nur Schnittstellennetze gemeldet.

## 12 Für das Sicherheitsteam des Kunden

Discovery soll gesehen und erkannt werden, nicht verborgen bleiben. Stimmen Sie Folgendes mit Ihrem Sicherheitsteam ab, bevor Sie ein Profil aktivieren:

- **Quelladressen:** die IP-Adressen der Hosts, auf denen ein Discovery-Agent läuft. Jeder Agent verbindet sich nur von seinem eigenen Host; CLM selbst baut nie Verbindungen in Ihr Netzwerk auf.
- **Ziele und Ports:** nur die genehmigten CIDRs ohne Ausnahmen; die Erreichbarkeitsports, dann die gängigen Ports aktiver Hosts und im Modus `full` jeder TCP-Port 1–65535 aktiver Hosts. Es handelt sich um gewöhnliche TCP-Verbindungen (vollständiger Handshake, keine Raw-Pakete, keine gefälschten Adressen) mit anschließendem TLS-Handshake; die Verbindung wird geschlossen, sobald das Zertifikat gelesen und der Dienst erkannt wurde.
- **Diensterkennung:** Der TLS-Handshake bietet die ALPN-Protokolle `h2` und `http/1.1` an (nicht auf Mail-, LDAP-, AMQP-, MQTT- und DNS-über-TLS-Ports) und erfasst TLS-Version, Cipher-Suite, ALPN-Protokoll und die vom Endpunkt vorgelegten Ausstellerzertifikate. Danach sendet der Agent an einen Endpunkt, der ein HTTP-Protokoll ausgehandelt hat oder keines auf einem HTTP-Port (443, 4443, 8443, 9443, 10443, 9000), genau eine Anfrage — `HEAD / HTTP/1.1` mit `Host` = Adresse, `User-Agent: CERTYX-Discovery`, `Accept: */*` und `Connection: close` — und liest nur Statuszeile und Header (höchstens 16 KiB, 2 Sekunden); er folgt keiner Weiterleitung, liest keinen Body und sendet nie ein `GET`, auch nicht nach `405`. Ein Endpunkt, der HTTP/2 ausgehandelt hat, erhält diese Anfrage über eine weitere Verbindung, die nur HTTP/1.1 anbietet. Auf SMTPS 465, IMAPS 993 und POP3S 995 sendet er nichts und liest die erste Begrüßungszeile. Andere Nicht-HTTP-Ports (25, 110, 143, 587, 636, 853, 2376, 3269, 5671, 5986, 6443, 8883) erhalten keine

Anfrage. Webserver- und IDS-Protokolle zeigen diese Anfragen mit dem User-Agent `CERTYX-Discovery`. `"Settings": { "ServiceIdentification": { "IdentifyServices": false } }` in der Scan-Konfiguration schaltet Anfrage und Begrüßungslesen ab; der Serverbetreiber tut dasselbe für serverseitige Scans mit `Clm:Discovery:ServiceIdentification:IdentifyServices`.

- **Reihenfolge und Zeitplan:** aufsteigende Adressen in fester Reihenfolge, keine Zufallsreihenfolge und kein Jitter; nur innerhalb der genehmigten Fenster in der Zeitzone des Profils; höchstens `ratePerSecond` neue Verbindungen pro Sekunde und Agent und höchstens 1–4 gleichzeitige Verbindungen pro Zielhost; automatische Verlangsamung, wenn viele Verbindungen in Zeitüberschreitungen laufen.
- **Agent als genehmigten Scanner registrieren:** Tragen Sie die Adressen der Agenten-Hosts in die Scanner- bzw. Schwachstellenscan-Ausnahmen Ihrer IDS/IPS, Firewalls, NAC- und EDR-Systeme ein, beschränkt auf die genehmigten CIDRs und Fenster, damit Alarme als erwartete Aktivität eingeordnet und nicht global unterdrückt werden. Ohne diese Registrierung wirkt vor allem der Modus `full` wie ein Portscan und kann blockiert oder als Angriff gemeldet werden.
- **Anhalten:** Das Deaktivieren des Profils in CLM beendet den Scan beim nächsten Segment; das Anhalten des Dienstes (`systemctl stop clm-discovery.service` oder der geplanten Aufgabe) beendet ihn sofort; mit `RestrictToAllowedCidrs` kann der Host-Eigentümer den Agenten unabhängig vom Profil auf lokal genehmigte Netze beschränken.
- **Nachweise:** Das Agentenprotokoll auf dem Host zeigt jedes Segment; CLM zeigt die Telemetrie des Agenten (Phase, Fortschritt, aktive Hosts, `waiting-window`, Drosselung).

## 13 Betriebseinstellungen

Jede Agenten-Konfigurationsdatei kann ein optionales `Settings`-Objekt auf oberster Ebene enthalten. Ausgelassene Werte behalten die unten genannten Standardwerte, die der Agent schon immer verwendet hat. Die Werte werden beim Start des Befehls geprüft; ein Wert außerhalb des zulässigen Bereichs beendet den Befehl vor jedem Netzwerkaufruf und wird nur mit dem Ausnahmetyp gemeldet (`InvalidDataException`). `enroll` schreibt kein `Settings`-Objekt; ein Administrator fügt es bei Bedarf der geschützten Konfiguration hinzu. Einstellungen enthalten niemals Anmeldedaten; bewahren Sie Geheimnisse in den in dieser Anleitung beschriebenen geschützten Dateien und Referenzen auf.

- `scan`-Konfiguration: `RequestTimeoutSeconds` 20 (1–300), `IdentityRenewalDays` 30 (1–179), `ConnectTimeoutMilliseconds` 500 (1–60000), `HandshakeTimeoutSeconds` 3 (1–300), `ProbeSpacingMilliseconds` 0 (0–10000, lokaler Mindestabstand zwischen Verbindungsaufbauten; 0 überlässt die Taktung der Rate des Profils, ein höherer Wert verlangsamt den Agenten nur), `SegmentSize` 200 (1–10000, Arbeitsschritte zwischen Profilprüfungen, Uploads, Telemetrie und Prüfpunkten), `IdleIntervalSeconds` 30 (1–86400, Pause, wenn kein Profil zugewiesen ist, und längste Wartezeit auf ein Scanfenster), `RetrySeconds` 30 (1–86400, Pause nach einem fehlgeschlagenen Zyklus), `NetworkMapIntervalMinutes` 60 (1–1440, Intervall der Netzwerkkarte), `RestrictToAllowedCidrs` false (`true`: jeder CIDR des Profils muss innerhalb der lokalen `AllowedCidrs` liegen), `BackOffFailurePercent` 50 (1–100), `BackOffRestorePercent` 25 (1–100), `BackOffMinimumPercent` 10 (1–100) sowie das Objekt `ServiceIdentification` mit `IdentifyServices` true (`false` beendet HEAD-Anfrage und Begrüßungslesen), `TimeoutMilliseconds` 2000 (100–60000), `MaximumResponseHeaderKilobytes` 16 (1–64), den Portlisten `HttpPorts`, `HttpRequestExcludedPorts`, `AlpnExcludedPorts` und `GreetingPorts` (je höchstens 256 Ports; Standardwerte in DISCOVERY-FULL.md, Abschnitt 8) und `UserAgent` `CERTYX-Discovery`. Modus, Ports, Rate, Verbindungen pro Host, Fenster, Versuche pro Zyklus und das Stapelintervall stammen aus dem genehmigten Profil.
- Konfiguration für `ssh manage` und `ssh monitor-krl`: `RequestTimeoutSeconds` 20 und `IdentityRenewalDays` 30, auf oberster Ebene der Datei (nicht innerhalb von `Identity`). Ihre Abfrageintervalle bleiben `PollSeconds` und `Subscription.PollSeconds`.
- Konfiguration des Lifecycle-Agenten: `PollSeconds` 10 (1–86400), `RequestTimeoutSeconds` 15 (1–300), `OperationTimeoutSeconds` 600 (1–86400, Obergrenze für einen Auftrag; ein Ziel in einem Cloud-Zertifikatsspeicher addiert das Dreifache seines Konvergenzfensters, bei AWS ACM 18 Minuten), `HeartbeatSeconds` 25 (1–119, Erneuerung der Lease; muss unter der serverseitigen Lease von 120 Sekunden und unter dem Operations-Timeout bleiben), `TrustChainTimeoutSeconds` 3 (1–300).
- Connector-Grenzen, in jeder Konfigurationsdatei einschließlich der lokalen `ssh`-Befehle (`renew`, `publish-krl`, `sync-krl`, `apply-krl` und die Befehle der Zertifizierungsstelle, die `Settings` aus der ersten Konfigurationsdatei lesen): `Settings.Connectors` kann `Process`, `Nginx`, `Deployment`, `HostDeployment`, `Acme`, `Dns`, `Vault`, `Secrets` und `CloudVerification` enthalten (Lifecycle-Agent und `secret-check`; `HostDeployment` enthält die Grenzen der dateibasierten Connectoren und des SSH-Relays: `CommandTimeout` 00:01:00 je Prüfung, Reload, Schlüsselerzeugung oder SSH-Durchlauf, `VerificationTimeout` 00:00:30, `RetryDelay` 00:00:00.5, `RevocationTimeout` 00:00:03, `MaximumReadBytes` 1048576 (4096–16777216) und `ChownExecutable` `/usr/bin/chown`; `CloudVerification` gilt für AWS-ACM-Ziele — wenn Sie es setzen, halten Sie `ConvergenceWindow` bei mindestens `00:06:00`, dem AWS-Standardwert bei ausgelassenem Objekt) sowie `SshKeygen`, `SshHostService`, `SshHostDeployment` und `SshKrlSubscription` (SSH-Befehle). Zeitdauern verwenden Text im Format `hh:mm:ss`, müssen positiv sein und höchstens eine Stunde betragen; Sekundenangaben für `SshHostService` liegen bei 1–300. CONFIGURATION.md listet alle Werte und Standardwerte unter „Customer-side connector limits“ auf. Ein ausgelassenes Objekt behält die Connector-Standardwerte.

Beispiel: `"Settings": { "SegmentSize": 100, "RetrySeconds": 60 }` oder `"Settings": { "Connectors": { "Nginx": { "VerificationTimeout": "00:00:30" } } }`. Protokoll- und Sicherheitsgrenzen sind fest: 5 Beobachtungen pro Upload, die in Schritt 4 beschriebenen Grenzen des Discovery-Bereichs, Begrenzungen der Antwortgröße, Journaleinträge von 1 MiB und RSA-3072-Identitäten mit 180 Tagen Gültigkeit.

## 14 Widerruf und Wiederherstellung

Verwenden Sie in CLM „Agent widerrufen“, um nachfolgende authentifizierte Aufrufe abzulehnen. Das Deaktivieren eines Profils stoppt nachfolgende Stapel; eine laufende Prüfung kann noch abgeschlossen werden. Die Lebensdauer einer Identität beträgt 180 Tage; Discovery-Agenten rotieren lokale Schlüssel und Identitäten automatisch 30 Tage vor Ablauf (Standardwert von `IdentityRenewalDays`). Für eine sofortige Rotation, zum Beispiel bei Verdacht auf einen offengelegten Schlüssel, führen Sie `Clm.Agent rotate-identity <config-file>` als Administrator aus (Linux: `sudo /opt/clm-agent/current/Clm.Agent rotate-identity /etc/clm-agent/agent.json`; Windows: `& "$env:ProgramData\CLM-Agent\Clm.Agent.exe" rotate-identity "$env:ProgramData\CLM-Agent\agent.json"`); der laufende Agent rotiert bei seiner nächsten Identitätsprüfung, und `status` zeigt danach den neuen Fingerabdruck. Bewahren Sie ausstehende Dateien während der Wiederherstellung auf. Eine Überlappung von 24 Stunden fängt verlorene Bestätigungen ab; siehe AGENT-IDENTITY-ROTATION.md. Der bestehende Lifecycle-Agent bleibt separat verfügbar und erfordert eine geprüfte CA- und Bereitstellungskonfiguration. Die

Netzwerkregistrierung aktiviert nicht dessen Bereitstellungsziel. Behalten Sie die lokalen Identitätsdateien, wenn Registrierungsbestätigungen unsicher sind; ersetzen Sie einen Schlüssel nicht blind. Eine fehlgeschlagene Einrichtung wird mit dem Ausnahmetyp und bei HTTP-Fehlern mit Statuscode oder Transportkategorie gemeldet (zum Beispiel `HttpRequestException (HTTP 401)` oder `HttpRequestException (SecureConnectionError)`), ohne Tokens auszugeben; eine abgelehnte Registrierung meldet den Status der Registrierung selbst, nicht den der Wiederherstellungsabfrage.

# 15 Paketerstellung

Betreiber erstellen die Pakete mit `bash scripts/package-agent.sh linux-x64` (oder linux-arm64, win-x64, win-arm64). Veröffentlichen Sie jede Architektur separat. Laufzeitpakete können Zugriff auf den NuGet-Feed von Microsoft erfordern. Setzen Sie `CLM_AGENT_VERSION` (major.minor.patch) und `CLM_RELEASE_SIGNING_KEY` (Pfad zum Offline-Release-Schlüssel ECDSA P-384), um das signierte Release-Manifest für Updates zu erzeugen. Mit den Produktionsverfahren `pkcs11` und `external` verweigert das Skript den Lauf ohne einen ECDSA-P-384-Schlüssel `CLM_RELEASE_SIGNING_KEY` und prüft die geschriebene Signatur; nur die Entwicklungsverfahren `pfx` und `unsigned-development` dürfen ein unsigniertes Manifest erzeugen, markiert mit „DEVELOPMENT PACKAGE", das Agenten als Update ablehnen (und Installer lehnen die fehlende Herausgebersignatur von `unsigned-development` ab, solange ihre Entwicklungsausnahme nicht gesetzt ist). Halten Sie beide Signaturschlüssel von den CLM-Servern fern. Die Code-Signierung des Herausgebers benötigt zusätzlich OpenSSL 3 und für Windows-Pakete `osslsigncode` (Authenticode unter Linux). Das Skript läuft erst, wenn mit `CLM_SIGN_BACKEND` ein Signaturverfahren gewählt ist; `scripts/sign-agent-package.sh` signiert:

- `pkcs11` (Produktion): Der Schlüssel bleibt in einem HSM, einer Smartcard oder einem Cloud-HSM mit PKCS#11-Modul. Setzen Sie `CLM_SIGN_PKCS11_MODULE` (Herstellermodul), `CLM_SIGN_KEY_URI` (`pkcs11:`-URI mit `pin-source=file:` auf eine nur für den Besitzer lesbare PIN-Datei; PIN-Werte in der URI werden abgelehnt) und `CLM_SIGN_CERT_CHAIN` (zuerst das Herausgeberzertifikat, dann Zwischenzertifikate). Die OpenSSL-PKCS#11-Engine (libp11) ist erforderlich.
- `external` (Produktion): `CLM_SIGN_AUTHENTICODE_COMMAND` (Aufruf `befehl <ein> <aus>`) und `CLM_SIGN_CMS_COMMAND` (Aufruf `befehl <ein> <aus.p7s>`, abgetrenntes DER-CMS mit SHA-256) kapseln Werkzeuge des Betreibers, zum Beispiel signtool auf einem Windows-Build-Host oder ein an einen Cloud-Schlüsseldienst gebundenes Werkzeug. Beide müssen dasselbe Herausgeberzertifikat verwenden.
- `pfx` (nur Entwicklung): `CLM_SIGN_PFX_FILE` und `CLM_SIGN_PFX_PASSWORD_FILE`, nur für den Besitzer zugänglich und außerhalb des Repositorys, sowie `CLM_SIGN_ALLOW_DEVELOPMENT_KEY=1`. Das Herausgeberzertifikat muss ein Code-Signing-Endzertifikat sein (`CA:FALSE`): Windows lehnt Authenticode-Signaturen eines CA-Zertifikats auch dann ab, wenn es vertrauenswürdig ist; `sign-agent-package.sh` bricht daher für Windows-Pakete mit einem solchen Zertifikat ab. `scripts/create-development-publisher.sh <new-directory>` erzeugt eine Entwicklungs-Herausgeber-CA und ein Endzertifikat mit PFX- und Passwortdatei; vertrauen Sie auf einem Windows-Entwicklungshost deren `publisher-ca.crt` (`Import-Certificate -FilePath publisher-ca.crt -CertStoreLocation Cert:\LocalMachine\Root`) oder übergeben Sie sie als `-PublisherCaFile`, niemals auf Kundenhosts.
- `unsigned-development`: Hash-Manifeste ohne Signaturen; Prüfer und Installationsprogramme lehnen solche Pakete ohne ihre ausdrückliche Entwicklungsausnahme ab.

Ein Windows-Build schreibt außerdem das Authenticode-signierte `verify-agent-package.ps1` neben die Archive; veröffentlichen Sie es mit ihnen, damit `Get-AuthenticodeSignature .\verify-agent-package.ps1` (der Befehl der Konsole) den festgelegten Herausgeber zeigt.

`CLM_SIGN_TIMESTAMP_URL` ergänzt Authenticode- und CMS-Signaturen um RFC-3161-Zeitstempel, damit Pakete nach Ablauf des Herausgeberzertifikats prüfbar bleiben. `CLM_SIGN_PUBLISHER_SHA256` prüft die Ausgabe vor dem erfolgreichen Abschluss erneut gegen den erwarteten Fingerabdruck. Der Release-Schritt gibt den Herausgeber-Fingerabdruck aus; veröffentlichen Sie ihn über Ihren unabhängigen Kundenkanal, nie nur neben dem Download. Signaturschlüssel, PINs und PFX-Dateien werden nie in das Paket oder die Build-Ausgabe geschrieben und dürfen nie eingecheckt werden. Veröffentlichen Sie das Archiv zusammen mit `.sha256`, `.sha256.p7s` (falls verwendet) `.p7s.tsr`, `.release.json` und `.release.json.sig`. Werden Container-Images ohne die signierten Archive gebaut (deploy/), kopieren Sie den vollständigen Paketsatz in das als `Clm:AgentPackages:Directory` konfigurierte Verzeichnis (die Bereitstellung bindet `$CERTYX_ROOT/agent-packages` ein); es wird unter `/downloads/` vor wwwroot/downloads ausgeliefert, auf dem Konsolen-Host und dem Agent-Host. Die Prüfer werden separat unter `/downloads/verify-agent-package.sh`, `/downloads/verify-agent-package.ps1` und `/downloads/verify-release.sh` bereitgestellt.

## 15.1 Reproduzierbare Builds, SBOM und Herkunftsnachweis

`scripts/build-agent.sh <rid> <verzeichnis>` erstellt die unsignierten Agent-Binärdateien reproduzierbar: im per Digest gepinnten .NET-SDK-Image aus `deploy/Dockerfile`, aus `git archive` des Commits, mit deterministischer Compiler-Ausgabe (`ContinuousIntegrationBuild`, abgebildete Quellpfade, der Commit als Quellrevision) und `SOURCE_DATE_EPOCH` gleich der Commit-Zeit. Es schreibt `clm-agent-<rid>.build.tar.gz` (sortiert, Besitzer 0, feste Zeiten), `clm-agent-<rid>.cdx.json` (CycloneDX-1.5-SBOM aller NuGet-Pakete und Runtime-Packs aus dem Restore-Ergebnis, ohne Netzwerkwerkzeuge) und `clm-agent-<rid>.build.json` (Version, Commit, SDK, Image und SHA-256 jeder Binärdatei). Zwei Builds desselben Commits ergeben identische Dateien. `scripts/package-agent.sh` verwendet dasselbe Skript (`--local`, mit dem SDK des Build-Rechners) und legt SBOM und Build-Nachweis in jedes Paket, wo `SHA256SUMS` sie abdeckt.

Der GitHub-Workflow `agent-release.yml` (Tag `agent-v<version>`) baut so alle vier Laufzeiten, bestätigt die Build-Herkunft (SLSA, `actions/attest-build-provenance`, wenn `CERTYX_ATTESTATIONS=true`) und hängt die Builds nach Freigabe in der Umgebung `agent-release` an ein Release-Entwurf an. Die Signaturschlüssel liegen nicht in GitHub. Vor dem Signieren prüfen:

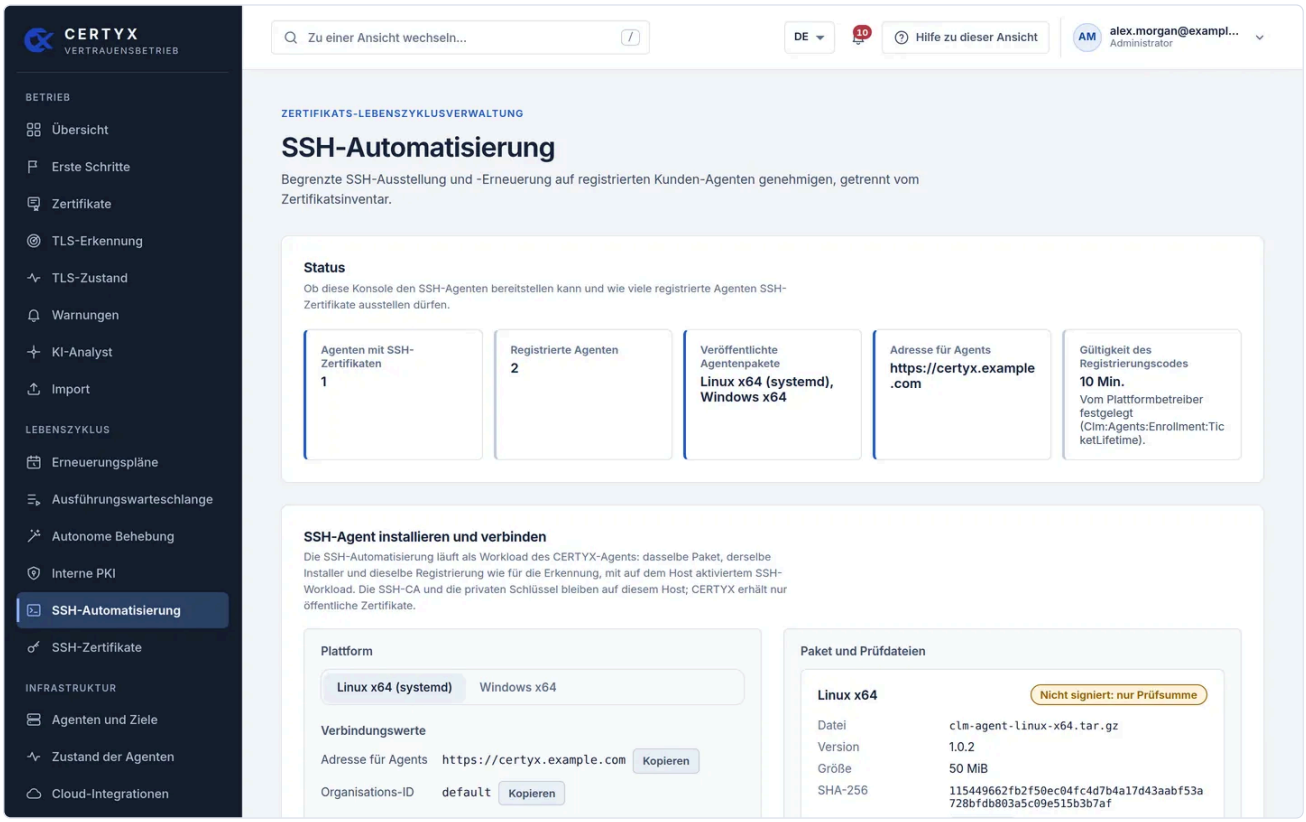
1. Herkunft: `gh attestation verify clm-agent-linux-x64.build.tar.gz --repo <owner>/<repository>` (ebenso für `.cdx.json` und `.build.json`).
2. Reproduzierbarkeit, unabhängig von GitHub: `git checkout agent-v<version>` und `CLM_AGENT_VERSION=<version> bash scripts/build-agent.sh linux-x64 /tmp/rebuild`; die `.build.tar.gz` mit der heruntergeladenen per `cmp` vergleichen (oder die `files`-Hashes beider `.build.json`).
3. Offline aus genau diesen Binärdateien paketieren und signieren: `CLM_AGENT_BUILD=<verzeichnis mit den drei Dateien> CLM_AGENT_VERSION=<version> CLM_SIGN_BACKEND=pkcs11 CLM_RELEASE_SIGNING_KEY=<schlüssel> bash scripts/package-agent.sh linux-x64`; das Skript lehnt Binärdateien ab, die von ihrem Build-Nachweis abweichen.

Auf einem Kundenhost nennt `clm-agent-<rid>.build.json` im Paket den Commit und den SHA-256 der Binärdatei (`sha256sum Clm.Agent`); Windows-Binärdateien tragen die nach dem Build hinzugefügte Authenticode-Signatur, daher weicht ihr Hash vom Build-Nachweis ab; Windows-Builds prüfen Sie über die `.build.tar.gz` des Release (Schritt 2).

Referenzen: https://learn.microsoft.com/en-us/dotnet/core/deploying/ und https://learn.microsoft.com/en-us/powershell/module/scheduledtasks/new-scheduledtaskprincipal

# 16 SSH

## 16.1 SSH-Workload: herunterladen, installieren und einrichten (Linux und Windows)




SSH-Automatisierung: SSH-Agent installieren und verbinden, verwaltete Profile und Freigaben (Beispieldaten)

Der SSH-Workload ist Teil dieses Agents; es gibt kein separates SSH-Paket. Wählen Sie in CERTYX → **SSH-Automatisierung** → **SSH-Agent installieren und verbinden** Linux x64 oder Windows x64: Der Bereich zeigt die Paketkarte dieser Plattform (Version, Größe, SHA-256 mit Kopieren-Schaltfläche, Signaturstand und Download-Schaltflächen für das Paket, seine `.sha256`, `.sha256.p7s` (wenn signiert) und den Prüfer; die Paket-Schaltfläche funktioniert in Ihrer angemeldeten Konsolensitzung), erstellt den Registrierungscode und zeigt die Befehle dieser Anleitung ausgefüllt (Adresse für Agents, Organisations-ID): Kopieren Sie clm-enrollment.token zuerst in ein leeres Arbeitsverzeichnis auf dem Server, denn der Paket-Download dort sendet den Code (siehe „Paket-Downloads“). Nach der Installation:

1. Erteilen Sie dem Agent in diesem Bereich **SSH-Zertifikate** (oder Agenten und Ziele → SSH-Vorgänge aktivieren); **SSH-Installation** nur, wenn der Agent Host- oder Benutzerzertifikate installieren soll.
2. Linux: `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh init` — Windows (Administrator-PowerShell): `powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\configure-ssh-windows.ps1" init`. Erstellt die SSH-CA des Hosts (`/etc/clm-ssh`, Eigentümer clm-agent, Modus 0700; `C:\ProgramData\CLM-SSH`, nur SYSTEM und Administratoren) und gibt `CA fingerprint: SHA256:…` aus.
3. Erstellen Sie das Profil mit diesem Fingerabdruck in der SSH-Automatisierung, lassen Sie es von einem zweiten Administrator genehmigen und laden Sie es herunter (**Profil herunterladen**, `ssh-profile-<ID>.json`).
4. `… configure-ssh-linux.sh add-profile ./ssh-profile-<ID>.json` bzw. `… configure-ssh-windows.ps1 add-profile .\ssh-profile-<ID>.json`: schreibt `managed.json`, führt einen geprüften Zyklus `ssh manage --once` aus und startet `clm-ssh.service` (Linux, Benutzer clm-agent) bzw. die geplante Aufgabe **CERTYX SSH Workload** (Windows, LocalSystem). Ein fehlgeschlagener Zyklus nennt seine Ursache, z. B. `HttpRequestException (HTTP 403)` ohne SSH-Berechtigung; es wird nichts gestartet.
5. Optional `trust-user-ca`: Der lokale sshd vertraut Benutzerzertifikaten dieser CA (`TrustedUserCAKeys`, mit `sshd -T` geprüft, Sicherung wiederhergestellt, wenn sshd nicht neu lädt). Unter Windows Server 2025 lässt die Standard-sshd_config nur die Gruppen `administrators` und `OpenSSH Users` zu (`AllowGroups`): Nehmen Sie Konten, die sich mit Benutzerzertifikaten anmelden, in `OpenSSH Users` auf (`Add-LocalGroupMember -Group 'OpenSSH Users' -Member <ACCOUNT>`), und starten Sie den OpenSSH-Server vor `trust-user-ca` einmal, weil er `sshd_config` beim ersten Start anlegt.
6. Prüfen: `sudo clm-agent status` (Windows: `Clm.Agent.exe status agent.json`) gibt `[ OK ] Capabilities: discovery, ssh; …` aus; `configure-ssh-linux.sh status` / `configure-ssh-windows.ps1 status` zeigen Dienst bzw. Aufgabe; die Tabelle der SSH-Automatisierung zeigt *SSH-Zertifikate: Erteilt* und den Bericht des SSH-Workloads.

Erkennung und SSH-Workload teilen die registrierte Identität; die Rotation wird über `identity-rotation.lock` serialisiert. Für die geprüfte Installation von Host-Zertifikaten unter Linux (**SSH-Installation** erteilt) erlaubt `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh enable-host-deploy` dem Konto clm-agent genau `sshd -t` und das Neuladen von sshd über sudo (mit visudo geprüfte Regel und Drop-in des Dienstes; `disable-host-deploy` entfernt beides). Details, Windows-ACL-Regeln und Ziele für die Host-Installation: SSH-LIFECYCLE.md.

Lokale SSH-Operationen sind unter Linux und Windows (OpenSSH für Windows) mit installiertem OpenSSH ssh-keygen verfügbar. Sie stellen Ed25519-Benutzer- und Host-Zertifikate aus und erzeugen eine KRL gemäß einer ausdrücklich genehmigten lokalen CA- und Principal-Richtlinie. Siehe SSH-LIFECYCLE.md und die Anleitung zum SSH-Arbeitsbereich. Von der Plattform genehmigte SSH-Aufträge und die verifizierte Rotation von Unix-Hosts sind über

eine separat autorisierte verwaltete Konfiguration verfügbar. Die Ausstellung allein installiert weder ein Benutzerzertifikat noch eine Vertrauensrichtlinie.

Die unbeaufsichtigte lokale SSH-Erneuerung ohne Meldung an CERTYX (`Clm.Agent ssh renew`) wird mit `sudo bash /opt/clm-agent/current/configure-ssh-linux.sh enable-renewal ./ssh-profile-<PROFILE-ID>.json` eingerichtet (Windows: `configure-ssh-windows.ps1 enable-renewal .\ssh-profile-<PROFILE-ID>.json`). Das Skript schreibt `renewal.json` aus der lokalen CA-Richtlinie und dem genehmigten Profil, führt einen Erneuerungszyklus aus und aktiviert `clm-ssh-renewal.timer` (jede Minute ein Dienst mit einem Zyklus unter `clm-agent`, ohne Netzwerkzugriff, Journal-Kennung `clm-ssh-renewal`; Windows: geplante Aufgabe `CERTYX SSH Renewal` als LocalSystem, Protokoll `state\ssh-renewal.log`). Ein Zyklus, in dem ein Profil fehlgeschlagen, blockiert oder im Richtlinienkonflikt ist, endet mit Code 3 und hinterlässt `clm-ssh-renewal.service` im Zustand failed (`systemctl --failed`); `renewal-status` zeigt den letzten Zyklus, `disable-renewal` entfernt Timer, Dienst und `renewal.json` (Zertifikate und Schlüssel bleiben). Ein Profil wird entweder mit `add-profile` (von der Plattform genehmigte Aufträge mit Meldung an CERTYX) oder mit `enable-renewal` erneuert, nie mit beiden. Profile, Wiederherstellung und Statuswerte: `SSH-LIFECYCLE.md`.

## 17 Vom Kunden kontrollierte Geheimnisse und KRL-Abonnements

Siehe CUSTOMER-VAULTS.en.md zu Referenzen auf Azure/AWS/GCP und HashiCorp Vault sowie zur Vorabprüfung mit secret-check. Siehe SSH-LIFECYCLE.md zu publish-krl/sync-krl, signierter Aktualität von einer Stunde, getrennten Relay-Berechtigungen und Wiederherstellung. Die Vorlagen examples/ssh-krl-publisher.json und examples/ssh-krl-subscription.json erfordern das Ersetzen aller Platzhalter und eine von null verschiedene Organisations-UUID. Eigenständiges sync-krl meldet Fehler nur lokal; den zentralen Status pro Abnehmer liefert `ssh monitor-krl` mit einer separat registrierten Identität, der die KRL-Abnehmer-Berichterstattung ausdrücklich erteilt wurde (sichtbar unter Zustand der Agenten).

## 18 SSH-Benutzeridentitäten für Workloads

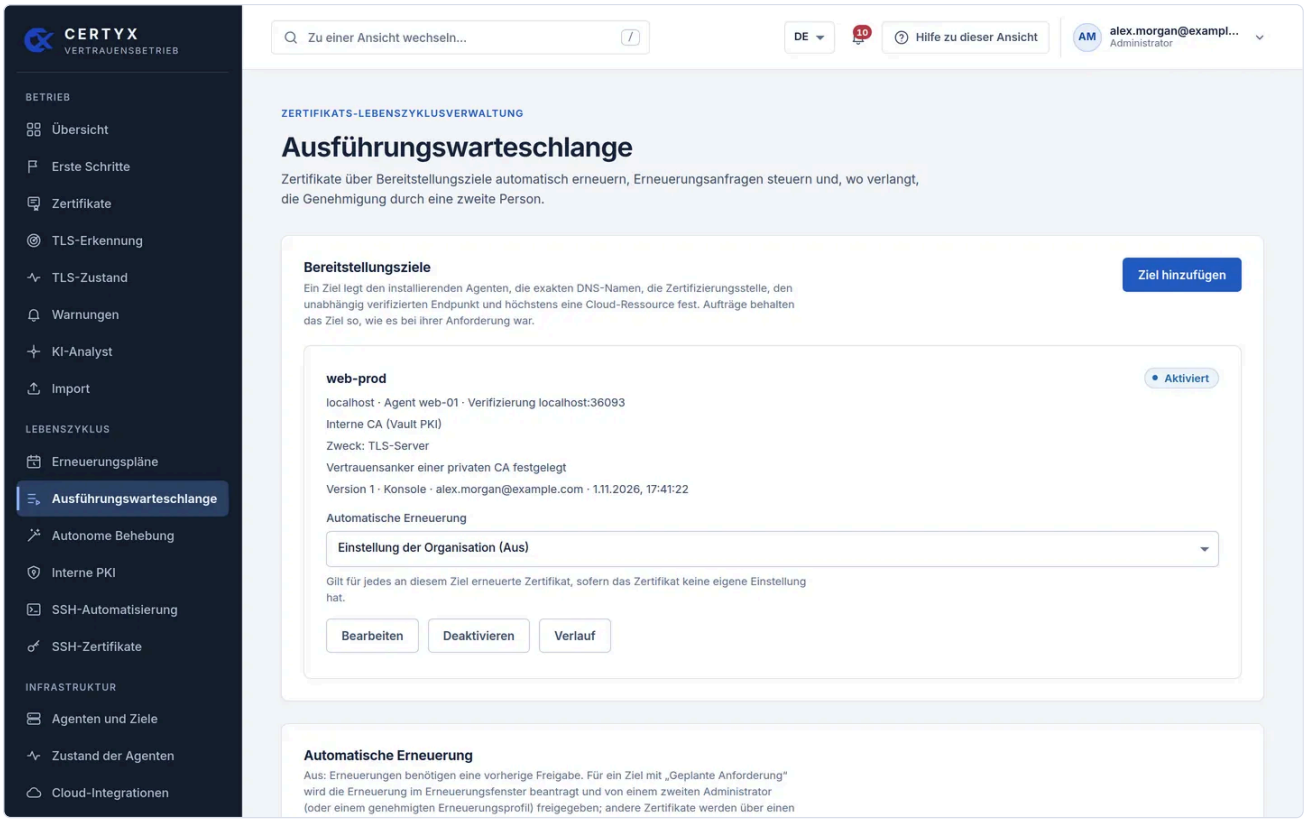
Aktuelle Unix-Agenten unterstützen außerdem genehmigte Benutzerprofile mit aktivierter Installation und passenden lokalen UserTargets. Siehe SSH-LIFECYCLE.md und examples/ssh-user-target.json. Der Agent läuft unter dem Workload-Konto, setzt ein vorhandenes, funktionierendes privates Bundle voraus, verifiziert die Zertifikatsauthentifizierung gegen ein separat fixiertes Serververtrauen und macht eine fehlgeschlagene Aktivierung rückgängig. Workloads müssen current/identity und current/identity-cert.pub verwenden. Aktualisieren Sie den Agenten, bevor Sie die Benutzerinstallation zuweisen; die Bereitstellung von Windows- bzw. persönlichen Konten und die Einrichtung des CA-Vertrauens auf entfernten Systemen bleiben separate Schritte.

## 19 Ziele für TLS-Client-Zertifikate

Ein Lifecycle-Ziel kann statt eines Server-Zertifikats ein TLS-Client-Zertifikat (Benutzer, Geräte, Dienste in Mutual TLS) erhalten. Setzen Sie `Purpose` auf `client` (oder `server-client` für beide Verwendungen; ohne Angabe gilt `server`) und listen Sie die Identitäten in `Client`: `Emails` (rfc822Name, z. B. `erika.mustermann@example.com`), `UserPrincipalNames` (z. B. `emustermann@corp.example.com`), `Uris` (z. B. `spiffe://example.org/ns/prod/sa/payments`) und optional `CommonName` (1–64 Zeichen; erforderlich, wenn nur URIs angegeben sind). `Domains` darf für ein Client-Zertifikat leer sein (`[]`). Die Werte müssen exakt dem Ziel in der Konsole entsprechen, sonst lehnt der Agent den Auftrag als außerhalb seines lokalen Bereichs ab. Client-Zwecke erfordern `AuthorityKind: "vault"` und eine Vault-Rolle mit `client_flag=true` (`server_flag=false` für reine Clients); ein Client-Ziel mit ACME oder DigiCert oder mit Cloud-Ressource wird beim Start mit diesem Grund abgelehnt. Ein reines Client-Ziel hat keinen Endpunkt: Lassen Sie `VerifyHost` und `Executable` leer oder setzen Sie `Executable` / `Arguments` auf einen festen absoluten Befehl (z. B. einen Dienstneustart), der nach der Aktivierung läuft. Der Agent schreibt `current/fullchain.pem` und `current/privkey.pem` (nur Eigentümer) und, wenn `Pkcs12PasswordReference` ein geschütztes Kundengeheimnis nennt, `current/bundle.p12` (PBES2 AES-256/SHA-256). Danach prüft er lokal, dass der installierte Schlüssel passt, das Zertifikat clientAuth erlaubt und die Kette zu `TrustRootsFile` aufgebaut werden kann; bei einem Fehler wird die vorherige Generation wiederhergestellt. Nur Unix; Client-Zertifikate werden weder in den Windows-Zertifikatsspeicher noch über ein SSH-Relay installiert.

```
{
    "Id": "vpn-jane",
    "Domains": [],
    "Purpose": "client",
    "Client": { "Emails": ["jane.doe@example.com"], "UserPrincipalNames": ["jdoe@corp.example.com"] },
    "DeploymentDirectory": "/etc/clm/client/vpn-jane",
    "KeyDirectory": "/var/lib/clm-agent/keys",
    "AuthorityKind": "vault",
    "VaultOrigin": "https://vault.example.com",
    "VaultMount": "pki_int",
    "VaultRole": "client-users",
    "VaultTokenFile": "/etc/clm/secrets/vault-token",
    "TrustRootsFile": "/etc/clm/trust/internal-roots.pem",
    "Executable": "",
    "Arguments": [],
    "VerifyHost": "",
    "VerifyPort": 0,
    "LifetimeHours": 720,
    "Pkcs12PasswordReference": { "Provider": "protected-file", "Location": "/etc/clm/secrets/vpn-jane-p12.password" }
}
```

# 20 Bereitstellungs-Connectoren und SSH-Relay




Ausführungswarteschlange: Bereitstellungsziele und Aufträge, die erneuerte Zertifikate installieren und jede Installation prüfen (Beispieldaten)

Ein erneuertes TLS-Server-Zertifikat wird in einem von zwei Modi installiert:

- **Agent auf dem Host** ( `local-agent` ): Der Lebenszyklusagent läuft auf dem Rechner, dessen Dienst das Zertifikat nutzt, erzeugt dort den Schlüssel und installiert dort die Dateien.
- **SSH-Relay** ( `ssh-relay` ): Ein Lebenszyklusagent installiert per SSH auf einem anderen Host, der keinen Agenten hat. Der Schlüssel wird auf diesem Host erzeugt und verlässt ihn nie; das Relay erhält nur den CSR.

Ohne Agent auf dem Host oder ein Relay mit SSH-Zugang zu ihm kann CERTYX ein Zertifikat nicht automatisch installieren; Discovery, Überwachung, Erneuerungspläne und Erinnerungen funktionieren weiterhin, und das Zertifikat wird dann von Hand installiert.

## 20.1 Connectoren

Ein Ziel nennt seinen Connector in `Connector` . Ohne `Connector` (oder mit `nginx` und weder `Files` noch `Remote` ) gilt unverändert das ursprüngliche Nginx-Layout: Generationen in `DeploymentDirectory` , `current/fullchain.pem` und `current/privkey.pem` sowie `Executable` / `Arguments` , gestartet mit `-t` und `-s reload` . Jeder andere Connector veröffentlicht die in `Files` genannten Dateien unter den Pfaden, die der Dienst bereits nutzt:

| Connector | Erforderliche Dateien | Standardprüfung | Standard-Reload | Modus der Schlüsseldatei | Hinweise |
|---|---|---|---|---|---|
| `pem-files` | beliebige der Dateiarten | keine | keiner | 0600 | Allgemein; setzen Sie `CheckCommand` und `ReloadCommand` für den Dienst. |
| `nginx` | `FullChain` + `PrivateKey` oder `Combined` | `/usr/sbin/nginx -t` | `/usr/sbin/nginx -s reload` | 0600 | `ssl_certificate` / `ssl_certificate_key` . |
| `apache` | `FullChain` + `PrivateKey` oder `Certificate` + `Chain` + `PrivateKey` | `/usr/sbin/apachectl -t` | `/usr/sbin/apachectl graceful` | 0600 | `SSLCertificateFile` (vollständige Kette seit Apache 2.4.8), `SSLCertificateKeyFile` , `SSLCertificateChainFile` für die separate Kette. |
| `haproxy` | `Combined` (vollständige Kette, danach der Schlüssel) | `/usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg` | `/usr/bin/systemctl reload haproxy` | 0600 | `bind :443 ssl crt <Combined>` . |
| `lighttpd` | `FullChain` + `PrivateKey` oder `Combined` | `/usr/sbin/lighttpd -tt -f /etc/lighttpd/lighttpd.conf` | `/usr/bin/systemctl reload lighttpd` | 0600 | `ssl.pemfile` und `ssl.privkey` (lighttpd 1.4.53 oder neuer). |
| `caddy` | `FullChain` + `PrivateKey` ; `Group` erforderlich | `/usr/bin/caddy validate --config /etc/caddy/Caddyfile` | `/usr/bin/caddy reload --config /etc/caddy/Caddyfile --force` | 0640 | `tls <FullChain> <PrivateKey>` im Site-Block; Caddy bezieht dieses Zertifikat dann nicht über sein eigenes ACME. |

| Connector | Erforderliche Dateien | Standardprüfung | Standard-Reload | Modus der Schlüsseldatei | Hinweise |
|---|---|---|---|---|---|
| `traefik` | `FullChain` + `PrivateKey` | keine | keiner | 0600 | File-Provider mit `watch: true`; legen Sie die Dateien in das überwachte Verzeichnis, damit die Umbenennung das Neuladen auslöst. |
| `envoy` | `FullChain` + `PrivateKey`; `Group` erforderlich | keine | keiner | 0640 | Dateibasiertes SDS (`path_config_source` mit `watched_directory` auf dem Zertifikatsverzeichnis); die Umbenennung löst die Rotation aus. Für SDS braucht die Bootstrap-Konfiguration `node` mit `id` und `cluster`. |
| `cribl` | `FullChain` + `PrivateKey`; `Group` erforderlich | keine | `/usr/bin/systemctl restart cribl` | 0640 | Verweisen Sie die Cribl-TLS-Einstellungen (API-Server oder eine Source) auf die Dateien (`certPath`, `privKeyPath`); Cribl Edge: `ReloadCommand` `["/usr/bin/systemctl","restart","cribl-edge"]`. Ein Ziel je Leader- oder Worker-Host. Ohne systemd `["/opt/cribl/bin/cribl","restart"]`. |
| `tomcat` | `Pkcs12` oder `JavaKeystore`; `Group` und `Pkcs12PasswordReference` erforderlich | keine | `/usr/bin/systemctl restart tomcat` | 0640 | `certificateKeystoreFile` mit `certificateKeystoreType="PKCS12"` und demselben Passwort; setzen Sie `ReloadCommand` auf Ihren Unit-Namen (zum Beispiel `tomcat10`). |
| `java-keystore` | `JavaKeystore`; `Pkcs12PasswordReference` erforderlich | keine | keiner | 0600 | keytool importiert das PKCS#12-Paket in `KeystoreType` `JKS` (Standard) oder `PKCS12` mit `KeystoreAlias`; setzen Sie `ReloadCommand`, um die Anwendung neu zu starten. |
| `postfix` | `FullChain` + `PrivateKey` | `/usr/sbin/postfix check` | `/usr/sbin/postfix reload` | 0600 | `smtpd_tls_chain_files = <PrivateKey>, <FullChain>`. Die Prüfung braucht implizites TLS: `VerifyPort` 465. |
| `dovecot` | `FullChain` + `PrivateKey` | `/usr/bin/doveconf -n` | `/usr/bin/doveadm reload` | 0600 | `ssl_cert = <<FullChain>` und `ssl_key = <<PrivateKey>` (Dovecot 2.3; 2.4 nutzt `ssl_server_cert_file` und `ssl_server_key_file`). Die Prüfung braucht implizites TLS: `VerifyPort` 993 oder 995. |
| `kubernetes-secret` | keine; Objekt `Kubernetes` | keine | keiner | — | `kubectl create secret tls … --dry-run=client` und `kubectl apply` auf dem Bereitstellungshost (`tls.crt` = vollständige Kette, `tls.key`); `VerifyHost` ist der Ingress oder das Gateway, das es ausliefert. Setzen Sie `ReloadCommand` (zum Beispiel ein `kubectl rollout restart`), wenn die Workload das Secret nicht beobachtet. |
| `windows-store` | keine; Objekt `WindowsStore` | — | optional `ReloadCommand` | — | Nur Windows-Agent: LocalMachine-Speicher, nicht exportierbarer Maschinenschlüssel. |
| `iis` | keine; `WindowsStore` mit der Bindung | — | `netsh http update sslcert` | — | Nur Windows-Agent; die HTTPS-Bindung muss existieren (einmal im IIS-Manager angelegt). |

Felder eines Connector-Ziels (alle aus der geschützten lokalen Konfiguration; ein Auftrag wählt nur das Ziel aus):

- `Files`: `Certificate` (Leaf), `Chain` (ausstellende Kette ohne Leaf), `FullChain` (Leaf und Kette), `PrivateKey`, `Combined` (vollständige Kette, danach der Schlüssel), `Pkcs12` (AES-256-PKCS#12) und `JavaKeystore`, jeweils ein absoluter Pfad aus Buchstaben, Ziffern und `. _ / @ + = , : % -`, in einem vorhandenen Verzeichnis, das nicht jeder beschreiben darf, ohne symbolische Links im Pfad, eindeutig und außerhalb des Generationsspeichers. `Owner` und `Group` setzen Besitzer und Gruppe jeder veröffentlichten Datei (der Agent oder das Relay-Konto muss den Besitz ändern dürfen, normalerweise root); `Mode` (Standard `0644`, kein Schreibrecht für Gruppe oder andere) gilt für öffentliche Dateien, `KeyMode` (Standard des Connectors, `0600` oder `0640`, nie für andere lesbar) für Dateien mit dem Schlüssel. `KeystoreType` (`JKS` oder `PKCS12`), `KeystoreAlias` (Standard `certyx`) und `KeytoolExecutable` (Standard `/usr/bin/keytool`, nur Agent auf dem Host) gelten für `JavaKeystore`.
- `CheckCommand` und `ReloadCommand`: Argument-Arrays, die mit einer absoluten ausführbaren Datei beginnen, zum Beispiel `["/usr/sbin/nginx","-t"]`; Argumente nutzen denselben Zeichensatz (keine Shell, keine Anführungszeichen, kein `$`, `;` oder `|`). Sie ersetzen die Standards des Connectors; `[]` schaltet einen Standard ab. Über SSH laufen sie hinter `Remote.SudoPrefix`.
- `Pkcs12PasswordReference`: die Kunden-Secret-Referenz des PKCS#12- und Keystore-Passworts (CUSTOMER-VAULTS.en.md); über SSH erreicht das Passwort den Remote-Host auf der Standardeingabe der verschlüsselten Sitzung, nie in einer Befehlszeile.
- `Kubernetes`: `KubeconfigFile`, `Namespace`, `SecretName`, optional `Context` und `KubectlExecutable` (Standard `/usr/local/bin/kubectl`, nur Agent auf dem Host).
- `WindowsStore`: `StoreName` (`My` oder `WebHosting`), `BindingHostName` (SNI-Bindung) oder `BindingAddress` (Standard `0.0.0.0`), `BindingPort` (443), `BindingAppId` (IIS: `{4dc3e181-e14b-4a21-b022-59fc669b0914}`) und `NetshExecutable`.
- `DeploymentDirectory`: auf dem Host des Agenten, privat (0700); enthält Bereitstellungsjournal und Sperre und, auf dem eigenen Host des Agenten, den Generationsspeicher. `VerifyHost` und `VerifyPort` nennen den TLS-Endpunkt, den der Agent nach dem Reload prüft (erforderlich).

Was der Agent für ein Connector-Ziel tut: Zuerst schließt er eine unterbrochene Bereitstellung ab; er legt den Speicher (0700) und auf dem Bereitstellungshost einen RSA-3072-Schlüssel mit CSR an (der CSR muss eine gültige Signatur und genau die genehmigten DNS-Namen tragen); lässt ihn von der CA signieren; prüft, dass das Zertifikat genau diesen öffentlichen Schlüssel und diese Namen trägt, TLS-Server-Authentifizierung erlaubt und zu `TrustRootsFile` aufbaut;

speichert eine neue Generation (`privkey.pem` 0600, `cert.pem`, `chain.pem`, `fullchain.pem`, die Pakete); veröffentlicht jede Datei atomar (eine neue Datei neben dem Ziel, dann eine Umbenennung) mit Besitzer, Gruppe und Modus; führt Prüfung und Reload aus; und prüft per TLS, dass `VerifyHost:VerifyPort` genau das neue Zertifikat präsentiert. CERTYX prüft den Endpunkt danach unabhängig, wie bei jedem Server-Ziel. Der Speicher behält die aktuelle und die vorherige Generation. Schlagen Prüfung, Reload oder eine Verifizierung fehl, wird die vorherige Generation erneut veröffentlicht, geprüft, neu geladen und verifiziert, und der Auftrag meldet `rolled-back`. Vor der ersten Bereitstellung kopiert der Agent die in diesem Moment veröffentlichten Dateien (Byte für Byte, mit Modus und Besitzer) als Ausgangszustand, den er wiederherstellt; existiert keine der Dateien, gibt es nichts wiederherzustellen, und der Agent stoppt mit „Intervention required“ und behält `journal.json` im Bereitstellungsverzeichnis: Prüfen Sie den Dienst und entfernen Sie dann das Journal. Ein Ausgangszustand, der nur eine `Combined`-, PKCS#12- oder Keystore-Datei hat, wird ohne Fingerabdruckprüfung wiederhergestellt, weil das Lesen seinen Schlüssel kopieren würde.

Beispiele: `examples/agent.deploy-<connector>.json` für jeden Connector und `examples/agent.ssh-relay.json`.

## 20.2 SSH-Relay

Das Relay ist ein registrierter Lebenszyklusagent mit **Ausführung des Zertifikatslebenszyklus aktivieren** (Berechtigung `deployment`). Jedes Remote-Ziel muss zusätzlich in `RelayDestinations` seiner Konfiguration stehen, und das Bereitstellungsziel muss den Installationsmodus **SSH-Relay** mit demselben Remote-Host nennen (Ausführungswarteschlange → Bereitstellungsziele → Installation); das Relay lehnt jeden Auftrag ab, dessen Ziel einen anderen Modus oder Host nennt. Voraussetzungen auf dem Remote-Host: OpenSSH-Server, eine POSIX-Login-Shell für das Relay-Konto (sh, bash, dash oder zsh), die üblichen Basiswerkzeuge (`mktemp`, `find`, `chmod`, `chown`, `mv`, `cp`, `sed`, `sort`), OpenSSL 1.1.1 oder neuer in `/usr/bin` oder `/usr/local/bin`, keytool für Java-Keystores, kubectl für Kubernetes und `ss` (iproute2) für `relay-check`. Nur TLS-Server-Zertifikate werden weitergegeben; Client-Zertifikate und der Windows-Speicher brauchen einen Agenten auf dem Host.

Die Schritte 2, 4, 5 und 6 auf dem Remote-Host sind ein Befehl des Agentenpakets: Kopieren Sie `configure-relay-host-linux.sh` (aus `/opt/clm-agent/current` auf dem Relay), den öffentlichen Schlüssel des Relays und den Helfer (Schritt 4) auf den Remote-Host und führen Sie als root aus:

```
sudo sh configure-relay-host-linux.sh enable --account certyx-deploy --public-key ./id_ed25519.pub \
  --helper ./certyx-relay-helper --from 10.0.0.5 --root /etc/haproxy/certs --root /var/lib/certyx-relay \
  --allow "/usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg" --allow "/usr/bin/systemctl reload haproxy"
```

Das Skript legt das Konto an (nur Schlüssel-Anmeldung), schreibt `authorized_keys` mit `restrict` und `from=`, installiert den Helfer, schreibt die roots-Datei, legt fehlende Verzeichnisse an, schreibt die sudoers-Regel für den Helfer und genau die `--allow`-Befehle (mit `visudo` geprüft) und gibt die festzulegenden Host-Schlüssel-Fingerabdrücke aus; `status --account certyx-deploy` zeigt die Einrichtung und `disable --account certyx-deploy` entfernt sie (veröffentlichte Zertifikate und Speicherverzeichnisse bleiben). Es arbeitet mit `useradd` (Debian, Ubuntu, RHEL, SUSE) und BusyBox `adduser` (Alpine). Die folgenden Schritte beschreiben, was es tut und wie es von Hand geht.

Schritt für Schritt (Beispiel: HAProxy auf `web-01.internal.example`, Relay-Konto `certyx-deploy`):

1. **Relay-Schlüssel** (auf dem Relay-Host, als root): `install -d -m 700 /etc/clm-agent/ssh` und `ssh-keygen -t ed25519 -N '' -C certyx-relay -f /etc/clm-agent/ssh/id_ed25519`. Der private Schlüssel bleibt nur für den Besitzer lesbar auf dem Relay; er wird nie an CERTYX gesendet.
2. **Remote-Konto** (auf dem Remote-Host): `useradd --system --create-home --shell /bin/sh certyx-deploy` und den öffentlichen Schlüssel in `~certyx-deploy/.ssh/authorized_keys` (Verzeichnis 700, Datei 600) mit Einschränkungen eintragen, zum Beispiel `restrict,from="10.0.0.5" ssh-ed25519 AAAA… certyx-relay` (die Adresse des Relays).
3. **Host-Schlüssel festlegen**: Lesen Sie auf dem Remote-Host den Fingerabdruck mit `ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub -E sha256`. Führen Sie auf dem Relay `ssh-keyscan -t ed25519 -p 22 web-01.internal.example > /etc/clm-agent/ssh/known_hosts` und `chmod 600 /etc/clm-agent/ssh/known_hosts` aus und vergleichen Sie `ssh-keygen -lf /etc/clm-agent/ssh/known_hosts -E sha256` mit dem Fingerabdruck vom Remote-Host. Tragen Sie diesen `SHA256:…`-Wert in `HostKeyFingerprints` ein. Der Agent lehnt eine known_hosts-Datei ab, die einen anderen Schlüssel für den Host, Platzhalter- oder negierte Einträge, `@cert-authority`- oder `@revoked`-Zeilen enthält; OpenSSH erzwingt sie dann mit strikter Host-Schlüssel-Prüfung (kein Vertrauen bei der ersten Verbindung). Für einen anderen Port lautet der Eintrag `[host]:port`.
4. **Relay-Helfer** (auf dem Remote-Host): Geben Sie ihn auf dem Relay aus dem signierten Agenten mit `sudo /opt/clm-agent/current/Clm.Agent relay-helper > certyx-relay-helper` aus, kopieren Sie ihn hinüber und installieren Sie ihn als root: `install -o root -g root -m 0755 certyx-relay-helper /usr/local/libexec/certyx-relay-helper`; vergleichen Sie `sha256sum` auf beiden Hosts. Der Helfer führt nur feste Operationen aus (schreiben, kopieren, öffentliche Dateien lesen, Speicherverzeichnisse anlegen und entfernen, Schlüssel und CSR erzeugen, PKCS#12, Keystore, Kubernetes-Secret, Listener) und findet seine Werkzeuge in einem festen PATH. Ohne `HelperPath` sendet das Relay dasselbe Skript über `/bin/sh -c`; das braucht keine Installation, lässt sich aber nicht über sudoers einschränken.
5. **Erlaubte Verzeichnisse**: `/etc/certyx-relay/roots` (Besitzer root, Modus 0644, ein absolutes Verzeichnis je Zeile) nennt, wo der Helfer schreiben und lesen darf, zum Beispiel `/etc/haproxy/certs` und `/var/lib/certyx-relay`. Die Datei ist Pflicht, wenn der Helfer als root läuft, und gilt auch ohne sudo; jeder Pfad muss unterhalb eines dieser Verzeichnisse liegen. Nennen Sie nur die Zertifikatsverzeichnisse und den Speicher, nie ein Web-Root oder ein Verzeichnis, das andere Konten beschreiben können. Der Helfer liest nur öffentliche Zertifikatsdateien und seinen Generationszeiger zurück, nie einen Schlüssel, eine kombinierte Datei oder einen Keystore.
6. **sudo** (wenn das Konto nicht root ist; bearbeiten mit `visudo -f /etc/sudoers.d/certyx-relay`): ``` Defaults:certyx-deploy !requiretty certyx-deploy ALL= (root) NOPASSWD: /usr/local/libexec/certyx-relay-helper, /usr/sbin/haproxy -c -f /etc/haproxy/haproxy.cfg, /usr/bin/systemctl reload haproxy ``` Nennen Sie genau den Helfer sowie die Prüf- und Reload-Befehle des Ziels mit ihren Argumenten. Setzen Sie `"SudoPrefix":["/usr/bin/sudo","-n"]`. Erlauben Sie weder `/bin/sh` noch einen Editor: Das gewährt volle root-Rechte.
7. **Ziel** in der Lebenszyklus-Konfiguration des Relays (`examples/agent.ssh-relay.json`): ```json "RelayDestinations": ["web-01-haproxy"], "Destinations": [{ "Id": "web-01-haproxy", "Domains": ["www.example.com"], "Connector": "haproxy", "DeploymentDirectory": "/var/lib/clm-agent/relay/web-01-haproxy", "Files": { "Combined": "/etc/haproxy/certs/www.pem" }, "Remote": { "Host": "web-01.internal.example", "Port": 22, "User": "certyx-deploy", "KnownHostsFile": "/etc/clm-agent/ssh/known_hosts", "HostKeyFingerprints": ["SHA256:..."], "IdentityFile": "/etc/clm-agent/ssh/id_ed25519", "StoreDirectory": "/var/lib/certyx-relay/web-01-haproxy", "SudoPrefix": ["/usr/bin/sudo", "-n"], "HelperPath": "/usr/local/libexec/certyx-relay-helper" }, "VerifyHost": "www.example.com", "VerifyPort": 443, "AuthorityKind": "vault", "...": "CA-Felder wie bei jedem Ziel" }] ``` Felder von `Remote`: `Host` (DNS-Name oder IP-Adresse), `Port` (22), `User`, `KnownHostsFile` und `IdentityFile` (absolute lokale Pfade, nur für den Besitzer, 0600), `HostKeyFingerprints`

(1–8 festgelegte `SHA256:`-Werte; für eine Schlüsselrotation beide angeben), `StoreDirectory` (auf dem Remote-Host, 0700: Schlüssel und Generationen), optional `JumpHost` (`Host`, `User`, `Port`, `HostKeyFingerprints`; erreicht mit derselben Identität und known_hosts-Datei), `ConnectTimeoutSeconds` (10, 1–120), `SshExecutable` (`/usr/bin/ssh`), `SudoPrefix` und `HelperPath`. `DeploymentDirectory` bleibt auf dem Relay und enthält nur Journal und Sperre.

8. **Prüfen, ohne ein Zertifikat zu ändern**: `sudo /opt/clm-agent/current/Clm.Agent relay-check <lifecycle-config.json> [destination-id]` prüft die Festlegung, die Authentifizierung, die Helfer-Version und das Speicherverzeichnis, nennt den Prozess, der auf `VerifyPort` lauscht, mit dem vorgeschlagenen Connector (in der Bereitschaft des Zertifikats als „Von relay-check gefundener Dienst“ angezeigt) und führt die Konfigurationsprüfung aus. Exit-Code 0 bedeutet bereit.
9. **Ziel in CERTYX**: Wählen Sie unter Ausführungswarteschlange → Bereitstellungsziele **Installation: Agent als SSH-Relay zu einem Host ohne Agent** und den Remote-Host (`Remote.Host`), mit dem Relay-Agenten als installierendem Agenten. Die Erneuerungsbereitschaft des Zertifikats zeigt dann das Relay, den Remote-Host und die veröffentlichten Dateien und blockiert mit einem klaren Grund, wenn Modus oder Host abweichen.

Das Relay verbindet sich mit dem OpenSSH-Client des Systems und nur mit Befehlszeilenoptionen (`-F /dev/null`): `BatchMode=yes`, `StrictHostKeyChecking=yes` gegen `KnownHostsFile`, Public-Key-Authentifizierung nur mit `IdentityFile` (`IdentitiesOnly=yes`, kein ssh-agent), keine Agent- oder X11-Weiterleitung, keine Port-Weiterleitung oder Tunnel, kein lokaler Befehl, keine gemeinsame Verbindung (`ControlMaster=no`) und `ConnectTimeout`. Jedes Remote-Wort steht in einfachen Anführungszeichen, sodass Pfade und Argumente wörtlich bleiben; Prüf- und Reload-Befehle laufen mit ihren Standardströmen auf /dev/null, sodass ein Neustart, der einen Daemon startet (zum Beispiel `cribl restart`), zurückkehrt, und nur der Exit-Code zählt. Jede Relay-Bereitstellung wird im Sicherheitsprotokoll festgehalten (`execution.relay.deployment.started|succeeded|failed|rolled-back` mit Agent, Ziel, Remote-Host und Namen). Ein Remote-Ziel sollte genau einem Relay gehören; zwei Relays, die dieselben Dateien bereitstellen, werden nicht koordiniert.

## 21 Zustellung gespeicherter Lifecycle-Ergebnisse

Gespeicherte Lifecycle-Ergebnisse werden nach vorübergehenden Verbindungsfehlern, Zeitüberschreitungen von Anfragen und HTTP 408/429/500/502/503/504 erneut übertragen, im Abstand von PollSeconds und begrenzt durch OperationTimeoutSeconds. Derselbe Auftrag, dieselbe Lease und dasselbe Ergebnis werden wiederverwendet; Ausstellung und Installation werden niemals wiederholt. Bei der Wiederherstellung nach einem Neustart erneuert der Agent die Lease, während er das gespeicherte Ergebnis zustellt. Fehler bei Authentifizierung oder Autorisierung, abgelaufene bzw. abgeglichene Leases und Fehler beim TLS-Vertrauen halten die Ausführung bis zum Abgleich an. Eine verlorene Lease kann durch diesen Mechanismus nicht erneuert werden; ein Ausfall, der länger dauert als die Lease oder das Operationslimit, erfordert weiterhin einen Abgleich durch den Administrator.

## 22 Geschütztes Lesen der Konfiguration

Alle JSON-Konfigurationsleser des Agenten lehnen vor der Deserialisierung symbolische Dateien oder übergeordnete Verzeichnisse, Unix-Berechtigungen für Gruppe/andere, Dateien über 1 MiB und ungültiges UTF-8 ab. Schützen Sie die Konfiguration unter Unix mit chmod 600. Relative Pfade in der Befehlszeile werden in einen absoluten Pfad aufgelöst und denselben Prüfungen unterzogen. UTF-8 mit oder ohne BOM wird unterstützt. Einstellungen und asynchrone Leser verwenden dieselben Regeln; Fehlerausgaben enthalten keine Konfigurationswerte. Unter Windows lesen dieselben Prüfungen die ACL: Eine private Datei oder ein privates Verzeichnis muss LocalSystem, den Administratoren oder der eigenen Identität des Agenten gehören, und kein anderer Principal darf sie lesen oder ändern (Verweigerungseinträge werden ignoriert). Dateien, die einem einzelnen Administratorkonto gehören, werden abgelehnt; das Installationsprogramm setzt die Administratoren als Besitzer.

## 23 Unabhängige Verifizierung im privaten Netzwerk

Verwenden Sie für Endpunkte, die nur innerhalb des Kundennetzwerks erreichbar sind, einen separat registrierten Endpunkt-Verifizierer. Wählen Sie unter Agenten und Ziele → Registrierte Agents für den Installer **Ausführung des Zertifikatslebenszyklus aktivieren** und für den Beobachter **Unabhängige Endpunktverifizierung aktivieren** (Fähigkeiten `deployment` und `endpoint-verification`). Verwenden Sie unterschiedliche Agentenidentitäten und private Schlüssel. Die Discovery-Registrierung allein erteilt keine dieser Berechtigungen. Registrieren Sie beide über den bestehenden HTTPS-Ablauf mit Einmal-Token; bewahren Sie jede erzeugte Identitäts- und Passwortdatei auf dem jeweiligen Kundenhost auf.

Ein Organisationsadministrator legt die ID des Verifizierungsagenten beim Kunden auf dem Bereitstellungsziel im Arbeitsbereich Ausführung (Bereitstellungsziele) fest, zusammen mit Verifizierungshost, Port und Health-Pfad; das API-Feld heißt `verificationAgentId`. Eine Änderung macht die bestehende Bereichsgenehmigung für die automatische Erneuerung ungültig. Lassen Sie die Verifizierer-ID weg, um die Verifizierung durch die Steuerungsebene beizubehalten. Der Remote-Modus erfordert einen registrierten Installer: Setzen Sie in dessen Lifecycle-Konfiguration sowohl `tenant` als auch `clientCertificatePasswordFile`. Die Passwortdatei hat Vorrang vor der veralteten Passwort-Umgebungsvariablen. Registrierte Lifecycle- und Verifizierer-Identitäten rotieren automatisch vor ihrem Ablauf.

**Prüfer installieren (ein Befehl).** Nennt das Bereitstellungsziel den Prüf-Agent, zeigt seine Karte unter Ausführung → Bereitstellungsziele **Unabhängigen Endpunkt-Prüfer einrichten**: Wählen Sie für diesen Agent **Unabhängige Endpunktverifizierung aktivieren**, laden Sie mit **Prüfumfang herunterladen** die Datei `endpoint-verifier-<TARGET-ID>.json` herunter (Ziel-ID, Prüfhost, Port, Health-Pfad und die Vertrauensanker des Ziels, nur öffentliche Daten), kopieren Sie sie auf den Prüfhost und führen Sie im Verzeichnis der Datei aus:

```
sudo bash /opt/clm-agent/current/configure-verifier-linux.sh add-target ./endpoint-verifier-<TARGET-ID>.json
```

```
powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\configure-verifier-windows.ps1" add-target .\endpoint-verifier-<TARGET-ID>.json
```

Das Skript schreibt die Prüfer-Konfiguration aus der eigenen registrierten Identität des Agenten und dem Prüfumfang (Linux: `/etc/clm-verifier/verifier.json` und `roots/<TARGET-ID>.pem`, Besitzer `clm-agent`, Modi 600 und 700; Windows: `C:\ProgramData\CLM-Agent\verifier`, Besitzer Administratoren, lesbar für LOCAL SERVICE), prüft sie mit den Bereichsregeln des Prüfers, führt einen geprüften Zyklus aus (`verify-endpoints --once`) und startet den Dienst: `clm-endpoint-verifier.service` unter dem Konto `clm-agent` (schreibgeschütztes System, `NoNewPrivileges`, nach einem Fehler und zusammen mit `clm-discovery.service` neu gestartet, sodass ein signiertes Update auch den Prüfer aus dem neuen Release startet) oder die geplante Aufgabe `CERTYX Endpoint Verifier` (LOCAL SERVICE, Protokoll `state\endpoint-verifier.log`). Solange CERTYX mit HTTP 403 antwortet, weil die Verifizierung noch nicht erteilt ist, wartet das Skript bis zu zehn Minuten auf die Erteilung (`CLM_VERIFIER_GRANT_WAIT` bzw. `-GrantWaitSeconds` in Sekunden; 0 bricht sofort ab und nennt den Konsolenschritt). Ein erneuter Lauf mit demselben Ziel ersetzt dieses Ziel; eine weitere Zieldatei fügt ein Ziel hinzu. `remove-target <TARGET-ID>` beendet die Prüfung eines Ziels (mit dem letzten Ziel wird der Dienst entfernt), `disable` entfernt Dienst, Konfiguration und Vertrauensanker, `status` zeigt den Dienst, seine letzten Protokollzeilen und den Agent-Status; `uninstall-linux.sh` und `uninstall-windows.ps1` entfernen alles mit dem Agent. Ein neuer Agent erhält die Rolle bei der Installation: Ergänzen Sie den Linux-Installationsbefehl um `CLM_AGENT_ENDPOINT_VERIFIER=./endpoint-verifier-<TARGET-ID>.json` (Windows: `-EndpointVerifierTarget .\endpoint-verifier-<TARGET-ID>.json`); das Installationsprogramm führt denselben Schritt nach der Registrierung aus, wartet auf die Erteilung und endet mit Code 3, wenn der Agent läuft, der Prüfer aber nicht aktiviert werden konnte.

Der Prüfer erreicht CERTYX mit dem Vertrauensspeicher seines Hosts (Vertrauensschritt des Installationsprogramms) und prüft den Endpunkt gegen die Vertrauensanker des Prüfumfangs; ein Prüfumfang ohne Vertrauensanker (ein Ziel, dessen Anker eine serverseitige Datei sind) benötigt die ausstellende CA im Vertrauensspeicher des Hosts. Der Produktivbetrieb erfordert HTTPS, vertrauenswürdige Ketten, Serverauthentifizierung, Online-Widerrufsprüfung und eine genaue Uhr. Eine von Hand aus `examples/endpoint-verifier.json` geschriebene Konfiguration (`identity` ist die Registrierungskonfiguration des Beobachters) folgt denselben Regeln; `testMode` deaktiviert die Online-Widerrufsprüfung und wird nur akzeptiert, wenn die Steuerungsebene und alle Ziele Loopback-Adressen sind; aktivieren Sie ihn niemals für Kunden-Workloads. Ausstellungs-Anmeldedaten, Installationspfade oder private Schlüssel von Workloads gehören nicht in die Prüfer-Konfiguration.

Nach der Installation bleibt die bestehende Lease aktiv, während der Installer nur sein gespeichertes Ergebnis erneut sendet. Ein separater Beobachter ruft eine 60-Sekunden-Anfrage ab, die an Organisation, Auftragsrevision, Lease, Ziel und erwarteten SHA-256-Fingerabdruck gebunden ist. Er validiert die live ausgelieferte TLS-Kette, den Endpunktnamen, die Server-EKU, den Widerrufsstatus, das exakte Zertifikat und die konfigurierte HTTP-Zustandsprüfung und gibt dann öffentliche Zertifikatsnachweise zurück. CLM akzeptiert einen Erfolg nur nach einem passenden Nachweis; ein fehlgeschlagener oder fehlender Beobachter kann keine erfolgreiche Bereitstellung herbeiführen. Der Beobachter kann seinen lokal genehmigten Zielbereich nicht aufgrund einer Serveranfrage erweitern. Selbstverifizierung und gemeinsam genutzte aktuelle/ausstehende Identitätsschlüssel werden abgelehnt.

Die Wiederherstellung der Ausführung verwendet eine neue Beobachtungsanfrage und behält die unabhängige Genehmigung sowie die optimistischen Revisionsprüfungen bei. Die Oberfläche wartet für einen begrenzten Zeitraum auf ausstehende Beobachtungen. Eine Zeitüberschreitung belässt den Auftrag zum Abgleich; Ausstellung oder Installation werden niemals stillschweigend wiederholt. Der Agentenzustand zeigt diese Workload separat an, einschließlich sicherer Fehler- und Zustände fehlender Meldungen. Eine fehlerfreie Erkennung kann einen fehlgeschlagenen Verifizierer nicht verdecken. Ermitteln Sie die Ausnahmekategorie im lokalen Dienstprotokoll und prüfen Sie Bereich, Stammzertifikate, Netzwerkerreichbarkeit, Health-URL, Uhr und Zuweisung der Fähigkeit; private Schlüssel und CA-Anmeldedaten sind niemals in der Telemetrie enthalten. Widerrufen oder entfernen Sie die Verifizierer-Fähigkeit, um weitere akzeptierte Beobachtungen zu stoppen.

### 23.1 Diagnose des Lifecycle-Installers

Registrierte Lifecycle-Agenten melden über ihre Bereitstellungsfähigkeit feste, sichere Gründe für Start, Abschluss und Fehler. Der Agentenzustand zeigt Deployment getrennt von der Erkennung an. Während aktiver Operationen wird der Zustand mit jedem Lease-Heartbeat aktualisiert und gibt das konfigurierte Heartbeat-Intervall an; Leerlaufzyklen verwenden das Abfrageintervall. Das Abfrageintervall registrierter Lifecycle-Agenten muss mindestens fünf Sekunden betragen. Eine fehlende Meldung wird nach Ablauf des Meldefensters zu einem Fehler, selbst wenn die Erkennung fehlerfrei bleibt. Ein Start oder ein abgeschlossener Zyklus beschreibt den Zustand des Prozesses und ist kein Nachweis, dass ein Zertifikat installiert wurde: Prüfen Sie den zugehörigen Ausführungsauftrag und den unabhängigen Endpunktnachweis. Unbekannte Nebenwirkungen von Ausstellung oder Installation halten die Ausführung weiterhin bis zum Abgleich an. Ein Meldefehler wiederholt keine Nebenwirkungen; fehlende Telemetrie wird vom Server erkannt. Ältere Installer aus der statischen Registry sind keine registrierten Agenten mit Bestandszustand.

### 23.2 Verwaltete Bereitstellungspfade

Lokale Verzeichnisse für die Bereitstellung von Schlüsseln und PEM-Dateien müssen absolute Pfade ohne Links haben, einschließlich ihrer übergeordneten Verzeichnisse. CERTYX legt Verzeichnisse an, die nur für den Besitzer zugänglich sind, und lehnt verlinkte oder ersetzte Stammverzeichnisse ab, bevor Schlüssel oder Bereitstellungsstatus geschrieben werden. Private Schlüssel und Journale müssen nur für den Besitzer zugänglich bleiben (Linux 600); erzeugte Verzeichnisse verwenden 700. Bei vorhandenen gemeinsam zugänglichen Journal- oder Sperrdateien müssen die Berechtigungen vor einem Neustart korrigiert werden. Die Wiederherstellung lehnt verlinkte, gemeinsam zugängliche oder übergroße Journale ab. Der einzelne relative Link `current` ist nur für eine verwaltete UUID-Generation zulässig; Generationsverzeichnisse und Zertifikatsdateien müssen selbst geschützt und linkfrei sein. Keine Konfiguration darf auf ein Benutzer-Home-Verzeichnis, einen gemeinsamen Upload-Bereich oder einen von außen beschreibbaren Bereitstellungsbaum verweisen. Diese Pfadprüfungen ersetzen keine Zugriffskontrollen des Hosts.

### 23.3 AWS ACM

Ziel für importierte AWS-Zertifikate: siehe AWS-LIFECYCLE.md und examples/agent.aws-vault.json. Ersetzen Sie alle Kennungen und Domains, registrieren Sie den Installer und den separaten Verifizierer, erteilen Sie die Bereitstellung ausdrücklich und genehmigen Sie ein Serverziel mit demselben AwsAcm-Bereich. Schützen Sie den vorhandenen Zertifikatsschlüssel bzw. die Referenz und bereiten Sie mit aws-adopt ein Rollback vor, bevor Sie Aufträge aktivieren. Das Beispiel verwendet die Ausstellung über Vault; es ist keine Vorlage für eine öffentlich vertrauenswürdige CA. AWS-Rollen-Anmeldedaten werden lokal vom SDK aufgelöst.

## 24 Öffentliche Zertifizierungsstellen: DNS-01, EAB und DigiCert

Ein ACME-Ziel kann die Validierung `dns-01` (`AcmeChallengeType`, für Wildcard-Zertifikate erforderlich) mit einem DNS-Anbieter in `AcmeDns` verwenden (Route 53, Azure DNS, Google Cloud DNS oder Cloudflare, beschränkt auf eine Zone) und sein Konto an einen Vertrag mit einer kommerziellen CA binden (`AcmeEabKeyId` und `AcmeEabHmacReference`). Ein `digicert`-Ziel bestellt oder stellt Zertifikate in DigiCert CertCentral im Rahmen der Kaufrichtlinie neu aus, und `Clm.Agent digicert-collect` schreibt einen Schnappschuss der gekauften Zertifikate zum Import in die Konsole. API- und HMAC-Schlüssel sind Geheimnisse des Kunden, die lokal aufgelöst werden. Details: CA-CONNECTORS.md.

### 24.1 Wiederherstellung der Discovery-Identität

Der Start der Erkennung führt nur lokale Validierung und Sperrung durch. Rotation und Anbieteraufrufe laufen im wiederholbaren Zyklus, sodass ein vorübergehender Ausfall beim Start den dauerhaft laufenden Agenten nicht beendet. Eine fehlgeschlagene Rotationsbestätigung markiert die Verbindung für den Neuaufbau, selbst wenn die nächste Rotation keine Änderung meldet; der alte Client wird erst freigegeben, nachdem ein Ersatz erfolgreich erstellt wurde. Fehlgeschlagene Telemetrie kann den behebbaren Zyklusfehler nicht verdecken. Zeitüberschreitungen von Anfragen und HTTP 408/429/500/502/503/504 werden nach Settings.RetrySeconds wiederholt; Fehler bei Authentifizierung/Autorisierung, Fehler beim TLS-Vertrauen und ein Abbruch durch den Benutzer beenden die Ausführung. `--once` endet bei einem fehlgeschlagenen Zyklus weiterhin mit einem Exit-Code ungleich null. Es werden keine Vertrauens- oder Bereichsprüfungen umgangen. Status- und Sperrpfade der Erkennung lehnen Links ab und erfordern einen Schutz nur für den Besitzer; ein vorhandener, breit zugänglicher Status wird abgelehnt, statt per chmod nachträglich zulässig gemacht zu werden.

## 25 Signierte Updates

Releases werden offline signiert: `scripts/package-agent.sh` schreibt `clm-agent-<rid>.release.json` (Version, Laufzeit, Archivname, Größe, SHA-256) und mit `CLM_RELEASE_SIGNING_KEY` zusätzlich dessen abgetrennte ECDSA-P-384/SHA-384-Signatur. Erzeugen Sie den Schlüssel einmalig auf einem Offline-Rechner: `openssl ecparam -name secp384r1 -genkey -noout -out release-signing.key` und `openssl ec -in release-signing.key -pubout -out release-signing.pub`; veröffentlichen Sie den öffentlichen Schlüssel und seinen Fingerabdruck (`openssl pkey -pubin -in release-signing.pub -outform DER | sha256sum`). CLM stellt diese Dateien nur unter `/downloads/` bereit und kann kein Update fälschen.

Wenn install-linux.sh den öffentlichen Schlüssel erhält, installiert es ihn als /etc/clm-agent-update/release-signing.pub (im Besitz von root), schreibt /etc/clm-agent-update/update.json (einschließlich `"AgentConfigurationFile":"/etc/clm-agent/agent.json"`) und aktiviert den täglichen `clm-agent-update.timer`. Der Update-Dienst (root) führt `Clm.Agent update /etc/clm-agent-update/update.json` aus, der:

- Manifest und Signatur über HTTPS herunterlädt und mit dem fixierten Schlüssel prüft; ein anderer Schlüssel, ein unsigniertes Manifest oder eine andere Laufzeit werden abgelehnt;
- nichts installiert, wenn die veröffentlichte Version gleich oder älter ist (kein Downgrade);
- das Archiv mit dem Client-Zertifikat und der Organisation des registrierten Agents (aus `AgentConfigurationFile`) herunterlädt, weil die Steuerungsebene Archive nur an berechtigte Anfragen ausliefert (siehe „Paket-Downloads“), Größe und SHA-256 mit dem signierten Manifest abgleicht und es ohne Links, absolute Pfade oder `..`-Einträge entpackt;
- die neue Binärdatei mit `--version` ausführt, die die signierte Version melden muss, dann /opt/clm-agent/current atomar umstellt, clm-discovery.service neu startet und auf `systemctl is-active` wartet;
- bei fehlgeschlagenem Neustart oder fehlgeschlagener Zustandsprüfung `current` wieder auf das vorherige Release setzt und dieses neu startet (Exit-Code 3); das vorherige Release wird immer aufbewahrt.

Felder von update.json: `ReleaseOrigin` (HTTPS), `Rid`, `ReleaseKeyFile`, `InstallRoot`, `RestartCommand` und optional `HealthCommand` (absolute ausführbare Dateien mit festen Argumenten), optional `TrustRootsFile` für eine private CA, `AgentConfigurationFile` (die `agent.json` des registrierten Agents, dessen Client-Zertifikat und Organisation den Archiv-Download berechtigen), `HealthTimeoutSeconds` (5–900, Standard 60) und `KeepReleases` (2–10). Exit-Codes: 0 installiert oder bereits aktuell, 1 abgelehnt ohne Änderung, 3 zurückgesetzt. Einer update.json eines älteren Installationsprogramms fehlt `AgentConfigurationFile`; diese Steuerungsebene beantwortet den Archiv-Download dann mit HTTP 401 und das Update installiert nichts. Führen Sie `sudo bash upgrade-linux.sh` aus einem neueren entpackten Paket aus, das die Eigenschaft ergänzt (siehe „Direktes Upgrade“), oder ergänzen Sie `"AgentConfigurationFile":"/etc/clm-agent/agent.json"` selbst in der root-eigenen /etc/clm-agent-update/update.json (Besitzer root, für andere nicht beschreibbar). Prüfen Sie mit `journalctl -u clm-agent-update.service`. Die Selbstaktualisierung läuft unter Linux; unter Windows prüft `upgrade-windows.ps1` dasselbe signierte Manifest (siehe „Direktes Upgrade“). Sie können das signierte ZIP auch selbst prüfen (PowerShell 7: PEM-Schlüssel mit `[System.Security.Cryptography.ECDsa]::Create()` und `ImportFromPem` importieren, dann `VerifyData` mit SHA-384 und `Rfc3279DerSequence`, oder OpenSSL verwenden). Ein aktualisiertes Release-Verzeichnis stimmt weiterhin mit `SHA256SUMS` des Pakets überein. Die Code-Signierung des Herausgebers beschreiben „Paketprüfung“ und „Paketerstellung“.

### 25.1 Direktes Upgrade

Eine neuere Version wird ohne neuen Registrierungscode über einen registrierten Agenten installiert; Konfiguration, Identität und Zustand bleiben unverändert. Braucht ein Schritt unten ein neueres Paket auf dem Host, laden Sie dessen Archiv über die angemeldete Konsole herunter (Download-Link unter Agenten und Ziele oder SSH-Automatisierung) und kopieren Sie es auf den Host, weil Archive nur an berechtigte Anfragen ausgeliefert werden (siehe „Paket-Downloads“); Prüfsummen, Signaturen und Release-Manifeste sind öffentlich.

- **Linux mit aktivierten signierten Updates** (mit Release-Schlüssel installiert): Der tägliche Timer aktualisiert automatisch; `sudo bash upgrade-linux.sh` (in einem neueren entpackten Paket) prüft sofort. Es ergänzt `AgentConfigurationFile` in einer vorhandenen update.json, der die Eigenschaft fehlt, sodass Installationen eines älteren Installationsprogramms wieder Archive herunterladen können. Außerdem darf der Erkennungsdienst danach die Netzwerke der Schnittstellen des Hosts lesen (`AF_NETLINK`), wenn ihn ein Installationsprogramm vor 1.0.1 eingerichtet hat.
- **Linux ohne Release-Schlüssel installiert:** Besorgen Sie release-signing.pub und seinen Fingerabdruck über einen vertrauenswürdigen Kanal von Ihrem Betreiber und führen Sie `sudo bash upgrade-linux.sh ./release-signing.pub` aus einem neueren entpackten Paket aus. Es fixiert den Schlüssel, schreibt /etc/clm-agent-update genau wie das Installationsprogramm (einschließlich `"AgentConfigurationFile":"/etc/clm-agent/agent.json"`, damit das Update

das Client-Zertifikat dieses Agents vorlegt), aktiviert `clm-agent-update.timer`, gibt den Fingerabdruck des Schlüssels aus und führt sofort `Clm.Agent update` aus. Ein anderer als ein bereits fixierter Schlüssel wird abgelehnt. Es braucht ein signiertes Release (`.release.json.sig`); ohne veröffentlichten Schlüssel gibt es keines, und das Upgrade bricht mit diesem Grund ab.

- **Windows, Download durch den Agenten** (empfohlen bei signierten Releases): Führen Sie in einer Administrator-PowerShell `powershell -NoProfile -ExecutionPolicy Bypass -File "$env:ProgramData\CLM-Agent\upgrade-windows.ps1" -Download -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub` aus (spätere Upgrades können `-ReleaseKeyFile` weglassen, der Schlüssel wird beim ersten Mal festgelegt). Das installierte Skript lässt `Clm.Agent.exe download-release` die Dateien `clm-agent-win-x64.release.json`, deren `.sig` und das Archiv mit dem Client-Zertifikat dieses Agenten von der Agentenadresse laden (kein Download über die Konsole, kein Kopieren), prüft Größe und SHA-256 des Archivs gegen das Manifest und führt dann alle Prüfungen und Schritte des nächsten Punkts aus. Ohne Release-Schlüssel (`-ReleaseKeyFile` oder ein bereits festgelegter) wird `-Download` abgelehnt. Ein erfolgreiches Upgrade mit Release-Schlüssel registriert außerdem die tägliche Update-Aufgabe `CERTYX Agent Update` (siehe Windows-Schritt 4), sodass spätere Releases ohne manuellen Lauf ankommen.
- **Windows**: Das Archiv `clm-agent-win-x64.zip` wird nur an berechtigte Anfragen ausgeliefert (siehe „Paket-Downloads“), und der Host ist bereits registriert, ein Registrierungscode kommt also nicht in Frage: Laden Sie das ZIP über die angemeldete Konsole herunter (Download-Link unter Agenten und Ziele oder SSH-Automatisierung) und kopieren Sie es auf den Host. `clm-agent-win-x64.release.json` und `clm-agent-win-x64.release.json.sig` sind öffentlich; laden Sie sie direkt von `/downloads/` der Agent-Adresse herunter. Entpacken Sie das ZIP und führen Sie in einer Administrator-PowerShell im entpackten Verzeichnis `.\upgrade-windows.ps1 -Archive ..\clm-agent-win-x64.zip -PublisherSha256 <PUBLISHER-SHA256> -ReleaseKeyFile .\release-signing.pub` aus (der Schlüssel wird beim ersten Mal in `C:\ProgramData\CLM-Agent-Update` fixiert; spätere Upgrades können `-ReleaseKeyFile` weglassen). Das Skript entpackt das Archiv erneut in das geschützte Programmverzeichnis, prüft Herausgebersignatur, signierte `SHA256SUMS` und Authenticode-Signaturen (wie das Installationsprogramm), danach prüft `Clm.Agent.exe verify-release` die Manifestsignatur gegen den fixierten Schlüssel, Größe und SHA-256 des Archivs und dass die Version neuer ist. Erst dann stoppt es die Aufgaben `CLM Discovery Agent` und `CERTYX SSH Workload`, verschiebt die ersetzten Programmdateien nach `C:\ProgramData\CLM-Agent\previous`, kopiert die neuen Dateien, verschiebt eine Identität älterer Installationen nach `identity\` (`Clm.Agent.exe migrate-identity`), startet die Aufgaben neu und verlangt, dass `Clm.Agent.exe status` innerhalb von 30 Sekunden erfolgreich ist. Andernfalls stellt es die vorherigen Dateien wieder her, startet sie neu und endet mit Code 3; eine abgelehnte Prüfung endet mit 1 und ändert nichts. Ohne veröffentlichten Release-Schlüssel lehnt das Skript ab, außer mit `-AllowUnsignedDevelopment` (nur Entwicklung: nur Herausgebersignatur und Versionsprüfung).

Die Windows-Identitätsrotation vor dieser Version konnte die nächste Identität nicht ablegen, weil das Programmverzeichnis für LOCAL SERVICE schreibgeschützt ist: Die Rotation stoppt dann mit „cannot write next to its identity“. Führen Sie `upgrade-windows.ps1` aus oder als Administrator `Clm.Agent.exe migrate-identity C:\ProgramData\CLM-Agent\agent.json` und danach `icacls C:\ProgramData\CLM-Agent\identity /grant *S-1-5-19:(OI)(CI)M`; die Rotation läuft dann weiter.

## 26 Prüfungen der Befehlszeile

`Clm.Agent check <https-origin>` (vor der Registrierung, ohne Identität) und `Clm.Agent status <config-file>` (nach der Registrierung, unter dem Agent-Konto; unter Linux `sudo clm-agent status`) geben pro Schritt eine Zeile `[ OK ]`, `[WARN]` oder `[FAIL]` aus und enden mit `Result:`; Exit-Code 0 bedeutet funktionsfähig, 1 ein fehlgeschlagener Schritt, 2 falsche Verwendung. Sie geben Hostnamen, Adressen, Zertifikatsantragsteller, den öffentlichen Zertifikatsfingerabdruck und Kontaktzeiten aus, nie Tokens, Passwörter oder Schlüssel.

Befehle von `Clm.Agent` (`Clm.Agent.exe` unter Windows); das erste Wort wählt den Befehl:

- `--version`: gibt die Release-Version aus.
- `check <https-origin>` und `status <config-file>`: die Prüfungen oben.
- `enroll <https-origin> <organisations-id> <token-datei> <config-file>`: einmalige Registrierung (siehe „Manuelle Ausführung / Ausführung im Container“).
- `scan <config-file> [--once]`: Netzwerkerkennung; diesen Befehl führen clm-discovery.service und die Windows-Aufgabe aus.
- `update <update.json>`: installiert ein neueres signiertes Release (Linux, siehe „Signierte Updates“).
- `verify-release <release-signing.pub> <release.json> <release.json.sig> <archive> <rid> [<installed-version>]`: die Offline-Prüfung eines heruntergeladenen signierten Releases (`VERIFIED`, Exit-Code 0, oder `REFUSED: <Grund>`, Exit-Code 1); verwendet von `upgrade-windows.ps1`.
- `migrate-identity <config-file>`: verschiebt eine Windows-Identität aus dem Programmverzeichnis nach `identity\` (siehe „Direktes Upgrade“).
- `rotate-identity <config-file>`: fordert eine sofortige Rotation der Identität an; der laufende Discovery-Dienst oder SSH-Workload ersetzt Schlüssel und Zertifikat bei seiner nächsten Identitätsprüfung (siehe „Widerruf und Wiederherstellung“).
- `download-release <config-file> <rid> <directory>`: lädt das veröffentlichte Release-Manifest, dessen Signatur und das Archiv mit dem Client-Zertifikat des Agenten in ein vorhandenes Verzeichnis; dabei wird nichts vertraut oder installiert (verwendet von `upgrade-windows.ps1 -Download`).
- `--log-file <path>` (mit jedem Befehl): schreibt die Ausgabe in eine Protokolldatei mit Zeitstempeln und höchstens 5 MB (plus `<path>.1`); die Windows-Aufgaben verwenden es.
- `<config-file> [--once]`: der Zertifikats-Lifecycle-Agent (jede Argumentliste, die zu keinem anderen Befehl passt).
- `verify-endpoints <config-file> [--once]`: der unabhängige Endpunkt-Verifizierer.
- `verifier add-target <agent-config> <verifier-directory> <target-file>` und `verifier remove-target <verifier-directory> <target-id>`: schreiben die Prüfer-Konfiguration (verwendet von `configure-verifier-linux.sh` und `configure-verifier-windows.ps1`).
- `ssh add-renewal-profile <ssh-directory> <profile-file>`: schreibt `renewal.json` der eigenständigen lokalen Erneuerung (verwendet von `enable-renewal`).
- `secret-check <config-file>`: Vorabprüfung der Kunden-Secret-Referenzen (CUSTOMER-VAULTS.en.md).
- `relay-helper`: gibt den Relay-Helfer dieses signierten Agenten zur Installation auf Hosts aus, auf denen ein SSH-Relay bereitstellt (siehe „SSH-Relay“).
- `relay-check <lifecycle-config> [destination-id]`: prüft die SSH-Relay-Ziele, ohne ein Zertifikat zu ändern: Host-Schlüssel-Festlegung, Authentifizierung, Helfer-Version, Speicherverzeichnis, den auf dem TLS-Port lauschenden Dienst mit dem vorgeschlagenen Connector und die Konfigurationsprüfung (Exit 0 bereit, 1 ein fehlgeschlagener Schritt, 2 falsche Verwendung).

- `aws-plan`, `aws-adopt`, `gcp-adopt`, `azure-adopt` und `digicert-collect`: Vorbereitung für Cloud und DigiCert (AWS-LIFECYCLE.md, GCP-LIFECYCLE.md, AZURE-LIFECYCLE.md, CA-CONNECTORS.md).
- `ssh manage|renew|monitor-krl|publish-krl|sync-krl <config-file> [--once]`, `ssh apply-krl <consumer-config> <envelope-datei>` und die lokalen Autoritätsbefehle `ssh issue|revoke|list|export-krl|deploy-host|deploy-user` (nur Unix, SSH-LIFECYCLE.md).

Das Linux-Installationsprogramm installiert das Hilfsprogramm `/usr/local/bin/clm-agent`: `sudo clm-agent status`, `sudo clm-agent check`, `sudo clm-agent log [Zeilen]` und `clm-agent version`. Unter Windows rufen Sie `Clm.Agent.exe` in `C:\ProgramData\CLM-Agent` direkt in einer Administrator-PowerShell auf.

Eine falsche Anzahl von Argumenten und unbekannte Optionen führen zum Abbruch mit `Usage:` und Exit-Code 2, bevor eine Konfiguration gelesen oder ein Verzeichnis angelegt wird: `secret-check`, `scan <config> [--once]`, `enroll`, `Clm.Agent <config> [--once]` und die SSH-Unterbefehle mit eigener Konfiguration (`manage`, `renew`, `monitor-krl`, `publish-krl`, `sync-krl`, `apply-krl`). Die Registrierung setzt voraus, dass die Tokendatei eine reguläre, nur für den Besitzer zugängliche Datei ohne symbolische Links im Pfad ist; dies wird geprüft, bevor das Token gelesen oder das Konfigurationsverzeichnis angelegt wird, und Registrierungsantworten sind auf 64 KiB begrenzt. Eine Lifecycle-Konfiguration, der ein erforderliches Element fehlt (Steuerungsebene, Client-Zertifikat, Statusverzeichnis, Ziele mit ID und Domains, Quelle des Zertifikatspassworts), bricht mit `InvalidDataException` ab. Der Discovery-Prüfpunkt wird nur für den Besitzer zugänglich und dauerhaft geschrieben. Vom Agenten gestartete Befehle (nginx, Neuladen von Diensten, ssh-keyscan, SSH-Prüfung) lesen niemals die Standardeingabe des Agenten; das SSH-Relay schreibt Dateiinhalte und Keystore-Passwörter nur auf die Standardeingabe seiner eigenen SSH-Sitzung.

## 27 Voraussetzungen des Windows-Hosts

Führen Sie vor der Installation `powershell -NoProfile -File .\check-windows.ps1` als Administrator im entpackten Paket aus. `certyx-windows-preflight.json` enthält ausschließlich Betriebssystem/Version/Architektur, PowerShell-Version, Administratorstatus und Verfügbarkeit von Aufgaben/SSH. Passwörter, Token, Zertifikatinhalte, Hostnamen und Benutzeridentitäten werden nicht erfasst. Der Bericht listet nur die Voraussetzungen auf; er installiert den Agenten nicht und prüft das Paket nicht. Prüfen Sie den freigegebenen Hash und die Signatur, bevor Sie Paketdateien ausführen. Der Installer verlangt einen expliziten HTTPS-Ursprung, verweigert verknüpfte Token-/Installationspfade und löscht seine lokale Tokenkopie auch bei einem Registrierungsfehler. Die ursprüngliche Tokendatei bleibt unter Ihrer Kontrolle; löschen Sie sie nach Gebrauch.