

Tanss Migration Project — Dokumentation

Projekt-Einleitung

Ibeco setzt für Fernwartung zunehmend RustDesk statt AnyDesk ein. Unser Ticket-/PSA-System TANSS kann Fernwartungs-Sessions aber ausschließlich über die AnyDesk-Schnittstelle abrufen. Das Tanss Migration Project schafft die technische Brücke, damit TANSS trotz des Wechsels zu RustDesk weiter nahtlos funktioniert, ohne dass TANSS selbst angepasst werden muss.

Beschreibung

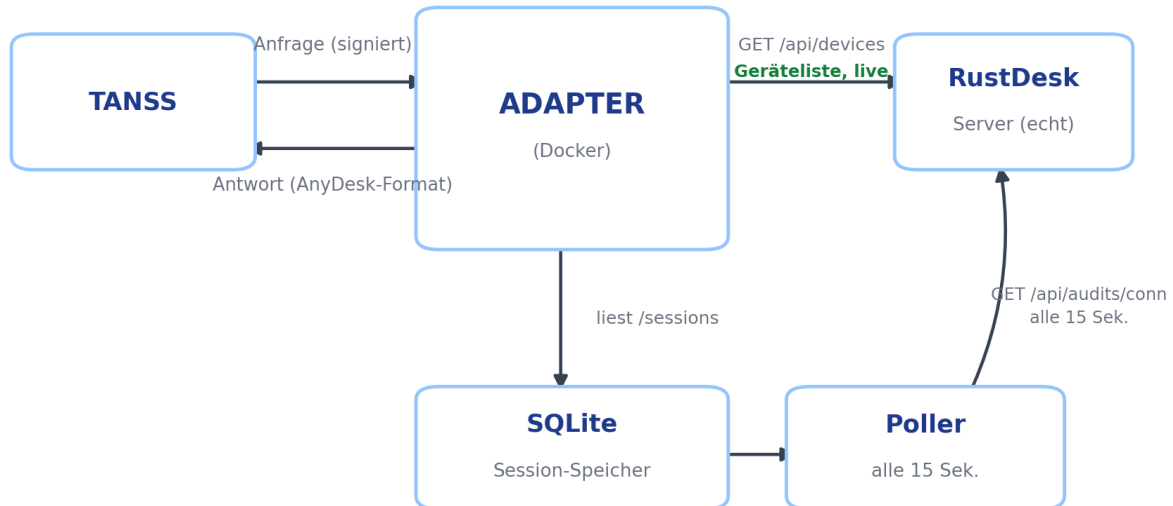
Dieses Dokument beschreibt den RustDesk↔AnyDesk-Adapter: einen Docker-Dienst, der sich gegenüber TANSS wie die echte AnyDesk-API verhält, im Hintergrund aber mit RustDesk spricht. Es behandelt Funktionsweise, Endpunkte, Netzwerk-Anbindung und Konfiguration (Umgebungsvariablen) für den produktiven Betrieb bei ibeco Pulheim.

1 Worum geht's?

TANSS kann Fernwartungs-Sessions nur von AnyDesk abrufen, es gibt dort keine RustDesk-Anbindung. Wir nutzen inzwischen aber RustDesk. Statt TANSS umzubauen, haben wir einen kleinen Docker-Dienst dazwischengeschaltet, der sich TANSS gegenüber exakt wie die echte AnyDesk-API verhält — TANSS merkt nicht, dass dahinter in Wirklichkeit RustDesk steckt.

2 Der Weg einer Anfrage

Zwei unterschiedliche Wege, je nachdem was TANSS abfragt — das ist der Teil, der auf den ersten Blick nicht offensichtlich ist:



■ **live** — Geräte (/clients, /sysinfo) werden bei jeder TANSS-Anfrage direkt und in Echtzeit bei RustDesk abgefragt.

■ **gespeichert** — Sessions (/sessions) kommen aus einer lokalen Datenbank, die ein Hintergrund-Prozess alle 15 Sekunden von RustDesk auffüllt.

3 Schritt für Schritt

01 TANSS fragt an

Genau wie bei echtem AnyDesk, mit einer signierten Anfrage (Lizenz-ID + Passwort, dieselben Werte wie früher bei AnyDesk).

02 Adapter prüft die Signatur

Stimmt sie nicht exakt, kommt 401 zurück — genau wie bei der echten AnyDesk-API.

03 Adapter holt die Daten

Geräte: sofort live bei RustDesk. Sessions: aus der eigenen Datenbank, die laufend im Hintergrund aktuell gehalten wird.

04 Adapter übersetzt die Antwort

RustDesk-Datenformat → exakt das JSON-Format, das TANSS von AnyDesk erwartet.

05 TANSS bekommt die Antwort

Sieht für TANSS aus wie eine normale AnyDesk-Antwort — TANSS weiß nicht, dass RustDesk dahintersteckt.

4 Die wichtigsten Endpunkte

ENDPUNKT	ZEIGT	QUELLE
GET /sysinfo	Lizenz- und Zähler-Übersicht	live + Datenbank
GET /clients	Geräteliste	live
GET /sessions	Fernwartungs-Verlauf	Datenbank
GET /health	Läuft der Dienst? (ohne Login)	—
GET /ready	Sind RustDesk + Datenbank erreichbar?	—

5 Gut zu wissen

■ **READ-ONLY** — Der RustDesk-Zugang ist bewusst nur lesend. Der Adapter kann bei RustDesk nichts verändern oder auslösen, nur abfragen.

■ **VON / ZU** — Techniker = „von“, Kunden-Gerät = „zu“. Bei jeder Session steht der Techniker (z. B. wer sich einwählt) auf der „von“-Seite, das Zielgerät des Kunden auf der „zu“-Seite, so wie TANSS die Zuordnung erwartet.

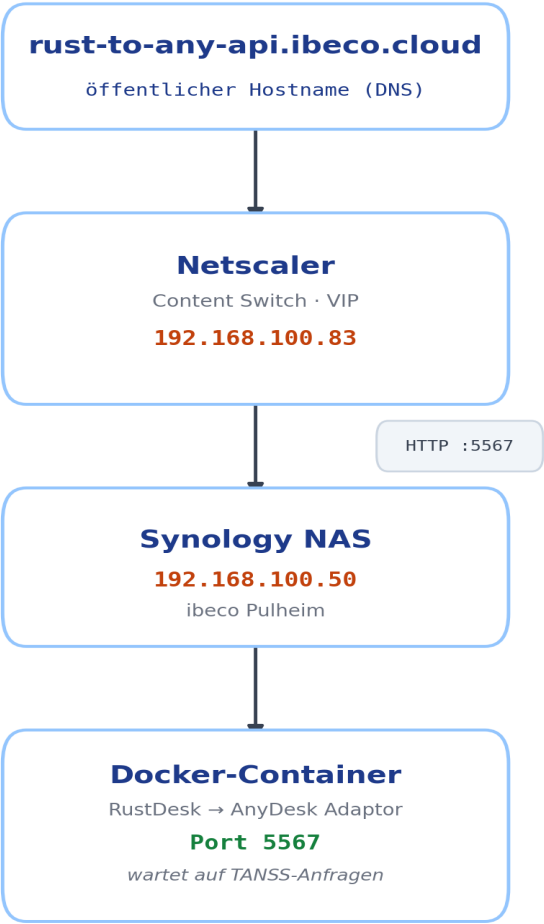
■ **NEUSTART-SICHER** — Die Session-Historie steht in einer Datenbank, nicht nur im Arbeitsspeicher — ein Neustart des Docker-Containers löscht nichts.

6 Betrieb

Läuft als Docker-Container bei ibeco Pulheim. Deployment über **Portainer** (Stack), das Image liegt auf **Docker Hub** und wird von dort automatisch gezogen — kein manuelles Kopieren von Dateien auf den Ziel-Server nötig.

7 Netzwerkweg

So kommt eine Anfrage von TANSS bis zum Docker-Container, die wichtigste Grafik in diesem Dokument:



Vier Stationen vom DNS-Namen bis zum Container: Netscaler-VIP → Content Switch → Synology NAS → Docker auf Port 5567.

Als Tabelle, mit der Rolle jeder Station:

STATION	ADRESSE	ROLLE
DNS-Name	rust-to-any-api.ibeco.cloud	Öffentlicher Einstiegspunkt, den TANSS anspricht
NetScaler Gateway (VIP)	192.168.100.83	Virtuelle IP, stellt den Content Switch bereit
Docker-Host (Synology NAS)	192.168.100.50:5567 (HTTP)	Der Content Switch leitet hierhin weiter, hier läuft der Adapter tatsächlich

8 Umgebungsvariablen

Der Docker-Container läuft bereits mit diesen 4 Werten (in Portainer beim Stack-Deploy gesetzt). Zwei davon sind Geheimnisse — die echten Werte stehen hier bewusst nicht drin, nur was sie sind und wofür sie gebraucht werden.

ANYDESK_LICENSE_ID

STEUERT

Welche AnyDesk-Lizenz-ID der Adapter TANSS gegenüber vorgibt und gegen welchen Wert die Signatur jeder TANSS-Anfrage geprüft wird.

ERWARTETER WERT

Die Lizenz-ID, die in TANSS unter „Fernwartung API-Anbindung“ hinterlegt ist.

WARUM NÖTIG

TANSS signiert jede Anfrage mit Lizenz-ID + Passwort, bei Abweichung antwortet der Adapter mit 401.

ANYDESK_API_PASSWORD

■ SECRET

STEUERT

Den zweiten Teil des Signatur-Schlüssels, mit dem der Adapter eingehende TANSS-Anfragen validiert.

ERWARTETER WERT

Das AnyDesk-API-Passwort, identisch zum in TANSS hinterlegten Wert. Nur in Portainer/Secrets hinterlegen, nie in Git oder dieser Doku.

WARUM NÖTIG

Ohne dieses Passwort kann der Adapter keine gültige Signatur berechnen und lehnt alle TANSS-Anfragen ab.

RUSTDESK_BASE_URL

STEUERT

Gegen welchen RustDesk-Server der Adapter seine Abfragen richtet.

ERWARTETER WERT

Vollständige URL des RustDesk-Servers, z. B. <https://remote.ibeco.cloud>.

WARUM NÖTIG

Ohne diese Adresse weiß der Adapter nicht, wo er Geräte- und Session-Daten abfragen soll.

RUSTDESK_TOKEN

■ SECRET

STEUERT

Authentifizierung gegenüber der RustDesk-API.

ERWARTETER WERT

RustDesk API Bearer-Token, read-only (Berechtigungen: Geräte + Audit Log). Nur in Portainer/Secrets hinterlegen, nie in Git oder dieser Doku.

WARUM NÖTIG

Ohne gültigen Token lehnt RustDesk jede Abfrage des Adapters ab.