Ordivis

TEIL I · GRUNDLAGEN

# Der Ordivis Stack

## Wie Ordivis Updater, Ordivis Service und die Fachanwendungen zusammengehören

Produktversion 2026.09.18 · Handbuch-Revision 1.0
Stand 18.09.2026 · Plattform .NET 10 / Windows Server

OU-01 Stack

# Der Ordivis Stack

Dieses Kapitel erklärt, wie eine Ordivis-Installation aufgebaut ist und welchen Platz **Ordivis Updater** darin einnimmt. Es richtet sich an **Administratoren**, die die Anlage betreiben.

Die Installationsschritte stehen in *Einrichten* (OU-10), der laufende Betrieb in *Aktualisieren* (OU-30), der Ernstfall in *Wenn ein Update schiefgeht* (OU-50).

INHALT



## 1 Was „Ordivis Stack" bedeutet

Eine Ordivis-Installation besteht nicht aus einem Programm, sondern aus **Schichten**. Jede setzt auf der darunter auf – wie beim OSI- oder TCP/IP-Modell. Für diesen Aufbau verwenden wir den Begriff **Ordivis Stack**.

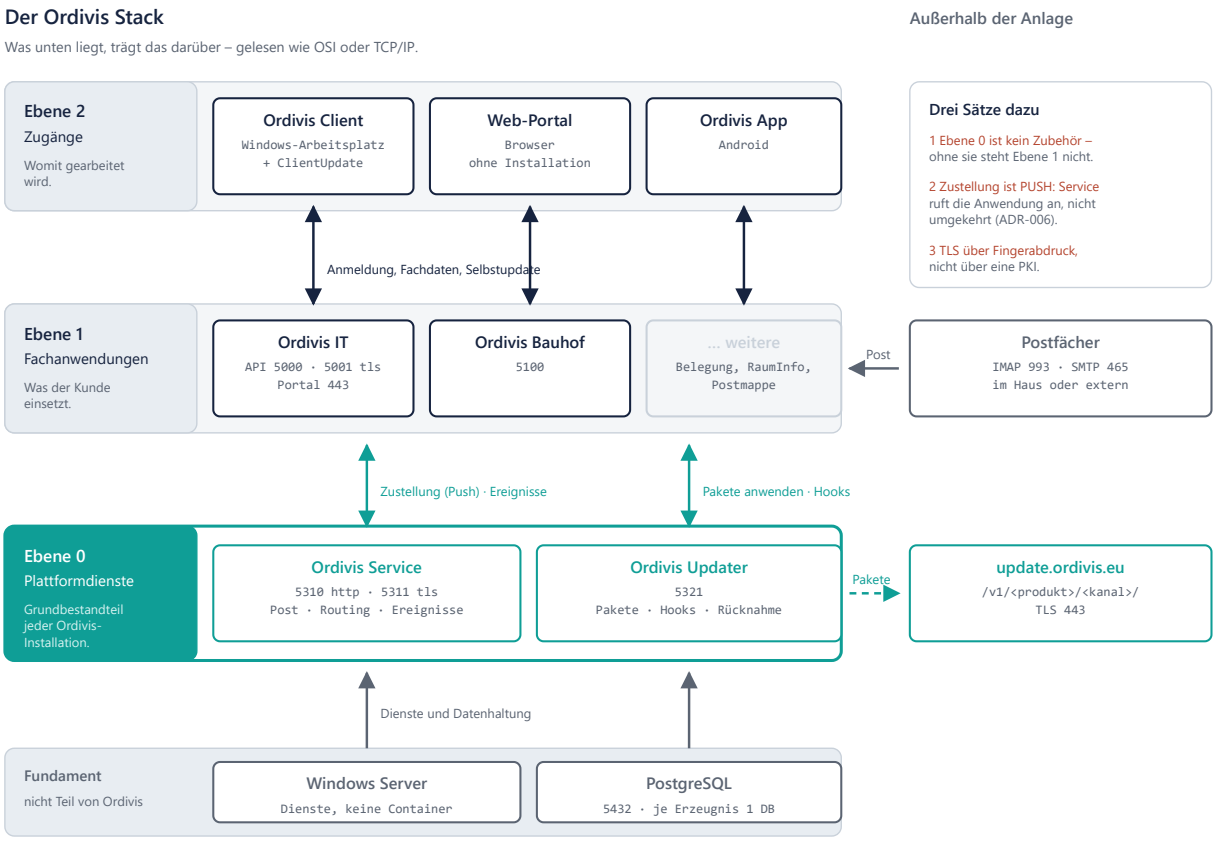


Abb. 1 – Der Ordivis Stack. Was unten liegt, trägt das darüber. Ordivis Updater bildet zusammen mit Ordivis Service die Ebene 0.

| Ebene | Was dort liegt | Beispiel |
|---|---|---|
| **2 – Zugänge** | Womit gearbeitet wird | Ordivis Client, Web-Portal, App |
| **1 – Fachanwendungen** | Was der Kunde einsetzt | Ordivis IT, Ordivis Bauhof |
| **0 – Plattformdienste** | Der gemeinsame Unterbau | Ordivis Service, **Ordivis Updater** |
| *Fundament* | Nicht Teil von Ordivis | Windows Server, PostgreSQL |

> **Ebene 0 ist kein Zubehör.** Ordivis Updater ist ein **Grundbestandteil jeder Ordivis-Installation**. Ohne ihn aktualisiert sich jede Fachanwendung selbst – mit eigenem Sicherungsverfahren, eigener Rücknahme und eigenem Zeitpunkt. Bei zwei Anwendungen ist das unübersichtlich, bei fünf nicht mehr zu verantworten.

## 2 Was Ordivis Updater tut

Er ist der **Wartungsdienst** der Anlage:

| Schritt | Was geschieht |
|---|---|
| **Bestand aufnehmen** | Welche Ordivis-Anwendungen liegen auf diesem Rechner, in welcher Fassung |
| **Katalog abgleichen** | Was der Hersteller anbietet |
| **Planen** | Was in welcher Reihenfolge zu tun ist – mit Abhängigkeiten |
| **Freigeben** | Ein Mensch bestätigt mit Namen; die Freigabe ist ein Nachweis |
| **Anwenden** | Sichern, Dienst anhalten, tauschen, migrieren, prüfen |
| **Zurücknehmen** | Wenn die Prüfung fehlschlägt – selbsttätig |

⚠️ **Inventar und Katalog sind zwei verschiedene Dinge, und die Unterscheidung ist wichtig.**

| | sagt |
|---|---|
| **Inventar** | „Ich *sehe* diese Installation" – die Bestandsaufnahme findet jede auf dem Rechner |
| **Katalog** | „Ich *kann* sie aktualisieren" – dafür braucht es ein signiertes Paket im Feed |

Eine Anwendung, die im Inventar steht, aber nicht im Katalog, kann der Updater **nicht** versorgen.

## 3 Wie eine Anwendung anschlussfähig wird

Ordivis Updater braucht keinen Code in der Fachanwendung. Sie liefert eine **Beschreibung** mit — `ordivis-hooks.json`, im selben Verzeichnis wie ihre Programmdatei:

```
{
  "productId": "bauhof",
  "phasen": {
    "pre":         [ { "art": "Datenbanksicherung", "verbindung": "haupt" } ],
    "migrate":     [ { "art": "Dateipruefung", "datei": "Ordivis.Bauhof.Api.dll" } ],
    "healthcheck": [ { "art": "Http", "adresse": "http://localhost:5100/gesundheit" } ],
    "rollback":    [ { "art": "Datenbankwiederherstellung", "verbindung": "haupt" } ]
  }
}
```

Drei Regeln, an denen es sonst scheitert:

1. ⚠️ `healthcheck` **ist Pflicht und muss fachlich prüfen** – Datenbank erreichbar, Schema auf Stand. „Prozess läuft" genügt nicht: Er ist die *einzige* Aussage, auf die sich der Updater bei „hat es geklappt?" stützt.
2. ⚠️ **Wer** `migrate` **zusagt, muss** `rollback` **zusagen.** Eine Anwendung, die vorwärts migriert und nicht zurückkann, macht den Rückweg des **ganzen Laufs** unmöglich.
3. ⚠️ **Die Sicherung gehört in** `pre`. Dort läuft die Anwendung noch, und es ist nichts angefasst — schlägt sie fehl, endet der Lauf, bevor er begonnen hat.

## 4 Die Verbindungen

| Von | Nach | Port / Adresse |
|---|---|---|
| Ordivis Updater | `update.ordivis.eu` | TLS 443, **ausgehend** |
| Administrator | Verwaltung Updater | 5321 |
| Arbeitsplatz-Client | Updater (Ringe, optional) | 5321 |
| Ordivis Updater | Dienste der Anwendungen | örtlich, kein Netz |

⭐ **Nur eine Verbindung verlässt die Anlage**, und sie geht **hinaus**. Von außen muss nichts hereinkommen.

## 5 Was Ordivis Updater nie anfasst

| Muster | Warum |
|---|---|
| `appsettings*.json` | Token, Anbindung, Verbindungszeichenfolgen |
| `*.key` | Schlüssel der Anlage |
| `data/**` | Nutzdaten |
| `server-url.txt` | Serveradresse, je Maschine |
| `*.db` | örtliche Datenbanken |

⚠️ **Diese Regel gilt für das Anwenden UND die Nachprüfung – als eine Regel, nicht als zwei Listen.** Als getrennte Listen führte das dazu, dass die Prüfung eine geschützte Datei für „fehlt" hielt und eine Rücknahme auslöste. Bei jedem Lauf, endlos.

## 6 Signaturen

Pakete sind **ES256-signiert**, und die Prüfung ist nicht abschaltbar. Jedes Produkt hat ein **eigenes** Schlüsselpaar; der Katalog nennt bei jedem Erzeugnis seine `keyId`.

⭐ **Ein Schlüssel für alles wäre ein Generalschlüssel.** Wer den Schlüssel eines Produkts erbeutet, kann damit gefälschte Pakete *dieses* Produkts unterschreiben – aber keine anderen.

⚠️ **Der private Schlüssel erreicht keine Anlage.** Er bleibt beim Hersteller. Auf der Anlage liegt ausschließlich der öffentliche Teil, unter `C:\ProgramData\Ordivis\Updater\schluessel\`.

Ordivis

TEIL IV · ERNSTFALL

# Wenn ein Update schiefgeht

## Was zu tun ist, wenn eine Anlage nach einem Update nicht mehr läuft

Produktversion 2026.09.18 · Handbuch-Revision 1.0

Stand 18.09.2026 · Plattform .NET 10 / Windows Server

OU-50 Notfall

# Wenn ein Update schiefgeht

Für den Moment, in dem eine Anlage nicht mehr läuft und jemand sie wieder zum Laufen bringen muss. Kurz gehalten und in der Reihenfolge, in der man vorgeht.

> **Zuerst, immer:** Die Anlage steht bereits auf dem letzten grünen Stand, wenn der Lauf `Zurückgerollt` heißt. Dann ist nichts kaputt – es ist nur nicht aktualisiert.

---

INHALT



## 0. Was ist überhaupt passiert?

`Verwaltung → Läufe`, den obersten öffnen. Der Zustand sagt alles Weitere:

| Zustand | Bedeutung | Was zu tun ist |
|---|---|---|
| **Erfolgreich** | durch, Gesundheitsprüfung grün | nichts |
| **Zurückgerollt** | ein Schritt scheiterte, die Rücknahme lief – die Anlage steht auf dem alten Stand | Grund lesen (§1), dann erneut versuchen |
| **Fehlgeschlagen** | ⚠️ die Rücknahme selbst kam nicht durch | §2 |
| **Abgebrochen** | endete, **bevor** etwas angefasst wurde | Grund lesen, nichts wiederherzustellen |
| **Übergeben** | Selbst-Update des Updaters läuft über den Bootstrapper | §3 |
| **Läuft** seit Stunden | ein Schritt hängt | §4 |

⚠️ **Der Zustand kommt aus dem Nachweis, nicht aus dem Gefühl.** Jeder Schritt steht mit Beginn, Ende, Ausgang und Ausgabe im Lauf. Bevor irgendetwas von Hand angefasst wird, lesen Sie dort, wie weit er gekommen ist – die Hälfte der Notfälle sind keine.

---

## 1. Zurückgerollt: die Anlage läuft, das Update nicht

Der Normalfall und die eingebaute Absicht. Die häufigsten Gründe:

**Die Gesundheitsprüfung der App blieb rot.** Das ist die einzige Aussage, auf die sich der Updater bei „hat es geklappt?" stützt. Sie prüft fachlich – Datenbank erreichbar, Schema auf Stand. Ein Dienst, der zwar startet, aber seine Datenbank nicht erreicht, ist zu Recht kein Erfolg.

**Ein Dienst ließ sich nicht anhalten.** Meist hält ein anderer Prozess eine Datei. Nachsehen mit:

```
Get-Process | Where-Object { $_.Path -like 'D:\Program Files\Ordivis*' }
```

**Die Sicherung fehlte.** Der Updater prüft nach `pre` nach, ob eine brauchbare Datenbanksicherung vorliegt (nicht abschaltbar). Fehlt sie, endet der Lauf, **bevor** etwas angefasst wird.

→ Ursache beheben, dann `Plan → freigeben → starten`.

---

## 2. Fehlgeschlagen: die Rücknahme kam nicht durch

Der ernste Fall. Hier ist die Anlage möglicherweise zwischen zwei Ständen.

1. **Nachsehen, welcher Schritt die Rücknahme verhinderte.** Es ist fast immer eine gehaltene Datei oder ein Dienst, der nicht anhielt.
2. **Die Sicherung liegt bereit.** Vor jedem Update sichert der `pre`-Hook die Datenbank, und die Sicherung wird **als Archiv geprüft**, nicht nur geschrieben. Pfad und Prüfergebnis stehen im Lauf.
3. **Wiederherstellen von Hand**, wenn die Rücknahme nicht nachzuholen ist: erst die Dateien aus dem Sicherungsordner, dann die Datenbank mit `pg_restore`.

⚠️ **Reihenfolge nicht tauschen.** Eine wiederhergestellte Datenbank unter neuen Programmdateien ist schlimmer als beides alt: Das Schema passt dann zu keiner Seite.

⚠️ `pg_restore --clean` **löscht nur, was in der Sicherung steht.** Hat das Update danach Tabellen, Sichten oder Fremdschlüssel angelegt, bricht das Zurückspielen an ihnen ab („kann Constraint … nicht löschen, weil andere Objekte davon abhängen"). Der automatische Rückweg räumt sie seit dem 18.09.2026 vorher selbst weg; ist er genau daran gescheitert, nennt seine Meldung sie einzeln („Zu entfernen war: …"). Diese Objekte zuerst entfernen, dann zurückspielen.

---

## 3. Übergeben: der Updater aktualisiert sich selbst

⚠️ **Dieser Zustand ist vorläufig, und das ist richtig so.** Ein Programm ersetzt seine eigenen Dateien nicht, während es läuft. Der Bootstrapper hält den Dienst an, tauscht und startet neu – der Lauf, der ihn gestartet hat, ist zu diesem Zeitpunkt tot und kann seinen eigenen Ausgang nicht mehr melden.

Beim nächsten Start schließt der Updater den Lauf ab, indem er die installierte Fassung gegen die Zielfassung hält.

**Ein Lauf, der in diesem Zustand liegen bleibt, sagt genau das Richtige: Der Dienst ist nicht wiedergekommen.** Dann:

```
Get-Service Ordivis.Updater
Start-Service Ordivis.Updater
```

Kommt er nicht hoch, liegt im Zielverzeichnis eine unvollständige Ablösung. Der Bootstrapper legt vor dem Tausch eine Kopie an; sie steht im Lauf.

⚠️ **Er läuft aus `%TEMP%`, nicht aus dem Installationsverzeichnis** – sonst hielte er die Datei fest, die er selbst ersetzen soll. Genau das ist am 05.09.2026 passiert und hat damals auch die Rücknahme verhindert.

---

## 4. Läuft seit Stunden

Jede Phase hat eine Zeitgrenze aus der `ordivis-hooks.json` der App (`pre`, `migrate`, `healthcheck`, `rollback`). Steht ein Lauf deutlich länger, hängt ein Hook.

`Verwaltung → Läufe → <Lauf>` zeigt den laufenden Schritt und seine bisherige Ausgabe. Ein `pg_dump` auf eine große Datenbank braucht seine Zeit – das ist kein Hängen.

⚠️ **Den Dienst nicht einfach beenden, solange ein Tausch läuft.** Dann bleibt das Verzeichnis zwischen zwei Ständen. Warten Sie die Zeitgrenze ab; der Updater bricht selbst ab und rollt zurück.

---

## 5. Nach einer Teilveröffentlichung: „das Produkt gibt es nicht mehr"

Meldet der Katalogabgleich ein Produkt als unbekannt, das es gestern noch gab, wurde beim Veröffentlichen eine **unvollständige Liste** geschrieben.

⚠️ **Der Katalog wird immer vollständig geschrieben, nie aus einer Teilmenge.** Wer nur ein Produkt veröffentlicht, aber die ganze Liste ersetzt, löscht die anderen – still, und es sieht aus wie ein verschwundenes Produkt.

→ Beim Hersteller vollständig neu veröffentlichen, dann `Verwaltung → Katalog → abgleichen`.

---

## 6. Wenn gar nichts mehr geht

Der Updater ist **nicht** der einzige Weg zurück. Jede Anlage lässt sich auch ohne ihn wieder in Gang setzen:

1. Dienste anhalten.
2. Programmdateien aus dem Sicherungsordner des letzten Laufs zurückspielen.
3. Datenbank aus der `pre`-Sicherung wiederherstellen.
4. Dienste starten, Gesundheitsprüfung der App von Hand aufrufen.

⚠️ **Was dabei nie überschrieben wird und nie überschrieben werden darf:** `appsettings*.json`, Schlüsseldateien, `data/`, `server-url.txt`, `debugConfig.json`, `*.db`. Dort stehen Token, Anbindung und maschinenspezifische Werte. Ein Rückspielen, das sie mitnimmt, nimmt der Anlage ihre Identität – sie läuft dann, gehört aber niemandem mehr.