Ordivis Belegung

BETRIEB

Öffentliche Auslieferung und Webserver

Der Spiegel nach außen, die Antragswarteschlange, der Webserver, Reverse Proxy und WAF

ALPHA · Fassung 0.9.0

Handbuch-Fassung 1.0 · Stand 29.08.2026

Grams IT · Produktreihe Ordivis · .NET 10 · PostgreSQL · Exchange

ID-51 Auslieferung

Öffentliche Auslieferung und Webserver

Achtung: Alpha-Fassung. Ordivis Belegung steht in der Fassung 0.9.0 und ist eine **Alpha-Fassung**. Die Anwendung wird laufend weitergebaut: **Masken, Menüwege, Bezeichnungen, Konfigurationsschlüssel und Abläufe können sich jederzeit und ohne Vorankündigung ändern** — und mit ihnen die Angaben in diesem Handbuch. Dieses Kapitel beschreibt den Stand vom 29.08.2026. Weicht die Anwendung von einer Beschreibung ab, gilt die Anwendung und nicht dieses Blatt; melden Sie die Abweichung. Für den Echtbetrieb ist diese Fassung **nicht freigegeben**.

Für den Betrieb **mit Trennung nach außen**: Anwendung im Hausnetz, öffentliche Seite auf einem Webserver im Internet.

Hinweis: Für den Anfang genügt der interne Teil. Läuft die Anwendung unmittelbar am Netz, ist dieses Dokument bis auf §6 (Reverse Proxy) und §7 (WAF) gegenstandslos.

INHALT

1 · Was draußen steht — und was nicht

Auf der Kundenmaschine draußen läuft kein .NET. Der öffentliche Webserver trägt nur:

Draußen	Inhalt
Statische Dateien	die Angular-Auslieferung
Momentaufnahme	der Belegungsspiegel als Dateien — **frei/belegt, ohne Personendaten**
ICS-Dateien	die Belegungspläne zum Abonnieren

Draußen	Inhalt
Ein PHP-Skript	`antrag.php` — der Antragseingang

Beide Pfeile gehen von innen nach außen (ID-02 §1). Der Webserver greift nie ins interne Netz.

> **Auf dieser Maschine liegt kein einziges Geheimnis.** Das Einzige, was dort steht, ist der **öffentliche** Schlüssel — er kann versiegeln und sonst nichts. Wer ihn hat, kann Anträge schreiben; **lesen** kann nur, wer die andere Hälfte hat, und die liegt drinnen.

2 · Veröffentlichung — der Schub nach außen

> ⚙️ **Am Server (innen):** `Veroeffentlichung:*` · 🔑 Schreibzugriff auf die Konfiguration

Schlüssel	Vorgabe	Inhalt
`Veroeffentlichung:TaktMinuten`	15	Wie oft die Momentaufnahme gebaut und geschoben wird
`Veroeffentlichung:Verzeichnis`	—	Örtliches Zielverzeichnis — für einen Betrieb **ohne** Trennung
`Veroeffentlichung:Sftp:Rechner`	—	Zielserver. **Gesetzt heißt: SFTP statt Verzeichnis**
`Veroeffentlichung:Sftp:Port`	22	
`Veroeffentlichung:Sftp:Benutzer`	—	Dienstkonto auf dem Webserver
`Veroeffentlichung:Sftp:SchluesselDatei`	—	Private Schlüsseldatei im OpenSSH-Format
`Veroeffentlichung:Sftp:SchluesselKennwort`	—	Nur falls die Datei eines trägt — **in die Geheimnisse**
`Veroeffentlichung:Sftp:Verzeichnis`	—	Wurzel auf dem Zielserver
`Veroeffentlichung:Sftp:Wirtsschluessel`	—	**Pflicht.** SHA256-Fingerabdruck des Zielservers

Der Wirtsschlüssel ist Pflicht

> **Achtung: Ohne ihn wird nicht geschoben.** Verschlüsselung sagt, dass niemand mitliest — sie sagt **nicht, mit wem gesprochen wird**. Ihn beim ersten Verbinden zu lernen, hieße, genau in dem Moment nichts zu prüfen, in dem es darauf ankäme.

Auf dem Zielserver abzulesen mit:

```
ssh-keygen -lf /etc/ssh/ssh_host_ed25519_key.pub
```

Kein Kennwort, nur Schlüssel

Ein Kennwort für einen Dienst, der sich alle 15 Minuten anmeldet, steht am Ende in einer Konfigurationsdatei. **Fehlt die Schlüsseldatei, weicht der Dienst nicht auf ein Kennwort aus** — er meldet es und schiebt nicht.

> **Achtung: Halb eingerichtet ist nicht eingerichtet.** Steht ein Rechner, fehlt aber eine der übrigen Angaben, meldet der Dienst das als **Fehler** und beendet sich.
>
> **Warum so streng:** Ein Schub, der still im Dateiverzeichnis landete, wäre schlimmer als keiner — draußen stünde ein alter Stand, und drinnen sagte niemand, warum.

3 · Warteschlange — der Antragseingang von außen

> ⚙️ **Am Server (innen):** `Warteschlange:*`

Schlüssel	Vorgabe	Inhalt
`Warteschlange:TaktMinuten`	1	Wie oft abgeholt wird
`Warteschlange:SchluesselDatei`	—	**Privater** Schlüssel, 32 Byte als Base64
`Warteschlange:Verzeichnis`	—	Örtliche Warteschlange (Betrieb ohne Trennung)
`Warteschlange:Sftp:*`	—	wie unter §2
`Warteschlange:HoechstgroesseKiB`	64	Größer wird nicht geladen, sondern beiseitegelegt
`Warteschlange:Antragsskript`	`/antrag.php`	Wohin der Assistent draußen schickt
`Warteschlange:AuskunftTage`	30	Wie lange die Eingangsauskunft nach außen geschoben wird

🔒 Pflichtreihenfolge

1. Schlüssel anlegen — **auf der internen Maschine**:

```
powershell pwsh -File tools\Neu-Warteschlangenschluessel.ps1 -Pfad C:\ProgramData\Ordivis\warteschlange.key
```

2. Anwendung starten. Sie meldet den **öffentlichen** Schlüssel:

```
[Start] Warteschlange=eingerichtet, oeffentlicher Schluessel 7bmGM73Qzz1cPo…
```

3. Diesen Wert in `antrag.config.php` **auf dem Webserver** eintragen und dort das Warteschlangenverzeichnis benennen — **neben** dem Web-Wurzelverzeichnis, nicht darin.

4. Prüfen, dass PHP `sodium` hat:

```
bash php -r 'var_dump(function_exists("sodium_crypto_box_seal"));'
```

5. Rechte setzen — das Verzeichnis gehört dem PHP-Benutzer und muss dem SFTP-Dienstkonto zugänglich sein.

✅ **Ergebnis:** Ein Testantrag von draußen liefert **202** und eine Eingangskennung; im Warteschlangenverzeichnis liegt eine `.umschlag`-Datei, in der **nichts Lesbares** steht.

Achtung: „Seit PHP 7.2 an Bord" heißt nicht „aktiv". Unter Debian und Ubuntu kommt die Erweiterung über `php-sodium` und ist üblicherweise eingeschaltet; auf Windows liegt `php_sodium.dll` bei, ist aber ohne `extension=sodium` **abgeschaltet**. Ohne sie bricht das Skript **beim ersten Antrag** ab — und zwar erst dann.

Der öffentliche Schlüssel wird gerechnet, nicht gepflegt. Zwei getrennt hinterlegte Hälften eines Schlüsselpaares laufen irgendwann auseinander — und dann geht kein Umschlag mehr auf, ohne dass jemand sagen könnte, welche der beiden falsch ist.

Warum eine Minute und nicht fünfzehn

Ein Antrag sperrt die Zeit als vorgemerkt, und der Zeitraum zwischen Antrag und Entscheidung ist genau der, in dem Doppelbuchungen entstehen. Über eine Warteschlange ist „sofort" nicht mehr zu halten — der Minutentakt drückt die Lücke auf eine Minute. **Ganz verschwindet sie nicht**, und der Zweite bekommt eine ehrliche Absage.

Unlesbare Umschläge

Achtung: Ein Umschlag, der sich nicht öffnen lässt, wird beiseitegelegt und nicht gelöscht — er bekommt die Endung `.unlesbar`.

Der häufigste Grund ist ein veralteter öffentlicher Schlüssel auf dem Webserver. Gelöscht wäre dann der Antrag einer Bürgerin weg, ohne dass ihn je jemand gesehen hätte.

Diese Dateien gehören überwacht (ID-52).

Die Eingangsauskunft

Sie ist der einzige Rückweg bei einer Absage. Ein angenommener Antrag wird zum Vorgang, und die Eingangsbestätigung führt den Link dorthin. Ein abgelehnter erzeugt **keinen Vorgang und keine Post** — ohne `/eingang/<kennung>` erführe die Antragstellerin nie, warum nichts kam.

Sie trägt **keine Vorgangsnummer**: Sie liegt als Datei auf dem öffentlichen Webserver.

Der Takt der Veröffentlichung bestimmt, wie schnell die Auskunft draußen ankommt (§2, Vorgabe 15 Minuten). Bis dahin steht dort „noch nicht eingelesen" — und genau das ist die ehrliche Auskunft.

Ist eine Warteschlange eingerichtet, geht der Antrag auch draußen — die Momentaufnahme trägt dann den Antragsweg. Ohne sie bleibt der Assistent draußen aus und sagt, warum: **Eine Datei nimmt keine Anträge an.**

4 · Die Weiche des Webservers

Achtung: `/oeffentlich` darf die Einseitenanwendungs-Weiche nicht verschlucken.

```
location /oeffentlich/ {
    try_files $uri =404;
}

location / {
    try_files $uri $uri/ /index.html;
}
```

Ohne die erste Regel bekommt die Oberfläche für jede fehlende Datei der Momentaufnahme eine `index.html` mit `200` — und liest HTML als JSON. Am sichtbarsten bei der Eingangsauskunft: Ein gerade abgeschickter Antrag ist noch nicht abgeholt, seine Datei gibt es also noch nicht. Mit der falschen Weiche steht dort nicht „noch nicht eingelesen", sondern ein Fehler.

5 · Die `index.html` darf kein Browser behalten

Achtung: Das ist keine Geschwindigkeitsfrage, sondern eine Funktionsfrage.

Die `index.html` verweist auf Bündelstücke mit einem Streuwert im Namen (`main-RMO5QF7N.js`). Bei jeder Auslieferung entstehen **neue** Namen, und die alten werden entfernt. Hält ein Browser die alte `index.html`, fordert er Stücke an, die es nicht mehr gibt — er bekommt `404`, und die Oberfläche fällt in sich zusammen.

So sieht es aus, wenn es passiert: *„Die Seite wird kurz angezeigt und verschwindet dann."* Es sieht aus wie ein Fehler der Anwendung und ist einer des Webservers.

Die Regel ist **umgekehrt zur Erwartung**: Das eine Dokument, das sich ständig ändert, wird nie behalten; alles Übrige, dessen Name sich mitändert, ein Jahr lang.

```
location = /index.html {
    add_header Cache-Control "no-cache, must-revalidate" always;
}

location ~* \.(js|css|woff2?|svg|png|jpg|webp)$ {
    add_header Cache-Control "public, max-age=31536000, immutable" always;
}
```

`no-cache` heißt nicht „nicht speichern", sondern „vor jeder Benutzung nachfragen". Mit `ETag` kostet das eine `304` und keinen erneuten Download.

Achtung: Die Regel gehört an jede Stelle, die die `index.html` ausliefert. Wer die Weiche getrennt konfiguriert hat — etwa einen eigenen Block für den Verwaltungsbereich —, braucht sie dort ebenfalls.

6 · Reverse Proxy

Nur nötig, wenn der Proxy auf einer *anderen* Maschine steht. Dann muss er unter `Proxy:Vertrauenswuerdig` eingetragen werden (ID-40 §10) — sonst sieht die Drossel einen einzigen Aufrufer, **und die erste Bürgerin sperrt die zweite aus**.

Was weitergereicht wird, steht in ID-02 §5.

Der Riegel gegen Massenversand steht im Webserver

Das PHP-Skript bringt Honigtopf und Zeitfalle mit — beide **ohne Barriere**, und beide ehrlich zu ihrer Grenze: Der Zeitpunkt kommt aus dem Browser und ist fälschbar. **Der Riegel ist `limit_req`:**

```
limit_req_zone $binary_remote_addr zone=antrag:10m rate=6r/m;

location = /antrag.php {
    limit_req zone=antrag burst=3 nodelay;
}
```

Dieselben Werte wie drinnen (ID-40 §9). Bei Plesk gehört das in die `vhost_nginx.conf` des Auftritts — **nicht** in die `nginx.conf` desselben Verzeichnisses: Die überlebt Plesks Neukonfiguration nicht.

7 · Web Application Firewall

Achtung: Prüfen Sie nicht nur, ob ein Antrag ankommt, sondern auch, ob eine *Absage* ankommt. Erfolgreiche Antworten sagen wenig über den Weg nach draußen aus.

Gemessen hinter Plesk mit dem Comodo-Regelsatz:

Fall	Antwort der Anwendung	Was beim Aufrufer ankam
Antrag angenommen	`201` mit Vorgangsnummer	`201`, unverändert
Termin belegt	`409` mit Ausweichvorschlägen	`409`, unverändert
Bestätigung fehlt	`400` mit dem Grund	`403 Forbidden – nginx`

Die Anwendung hat in allen drei Fällen richtig geantwortet. Eine Regel hat die dritte Antwort **verworfen und eine eigene an ihre Stelle gesetzt**. Für die Bürgerin heißt das: Sie hakt den Datenschutzhinweis nicht an, klickt „Absenden" und liest „403 Forbidden" — nicht den einen Satz, der ihr sagt, was zu tun ist. Und die Verwaltung sucht den Fehler danach in den Zeitregeln.

Die Regel gehörte dort nicht her: Sie suchte nach einer Lücke in *Eclipse Jetty vor 9.2.9* (CVE-2015-2080, aus dem Jahr 2015). Hier läuft ASP.NET Core.

> **Ein erster Verdacht war falsch, und das ist der Teil, der Zeit kostet:** Die Absage kam ursprünglich als reiner Text, die übrigen Antworten als JSON — das sah nach der Ursache aus. **Sie war es nicht.** Auch als ProblemDetails-JSON wurde die `400` verworfen. Die Regel stört sich an der **Statuszeile**, nicht am Inhalt. Wer nur das Format ändert, glaubt es behoben zu haben und liefert aus.

Behoben in der `vhost_nginx.conf` des Auftritts — **eine** Regel, **ein** Auftritt, der übrige Regelsatz bleibt in Kraft:

```
modsecurity_rules '
    SecRuleRemoveById 243420
';
```

Danach `plesk sbin httpdmng --reconfigure-domain <domain>`, `nginx -t`, `systemctl reload nginx`.

Bei einem anderen Regelsatz kann es eine andere Nummer sein. Der Weg zur Nummer:

```
tail -n 50 /var/log/nginx/error.log | grep ModSecurity
```

8 · Örtlich prüfen, ohne Server der Gemeinde

```
php -d extension=sodium -S 127.0.0.1:8791 -t webserver
```

Ein Antrag mit gültigem `begonnen` (Millisekunden, mindestens drei Sekunden alt) muss **202** und eine 43 Zeichen lange Eingangskennung liefern.

Verwandte Themen

- **ID-02** — Architektur: warum EWS nicht vom Webserver gelesen wird
- **ID-40** — Konfiguration: Missbrauchsschutz und Proxy
- **ID-52** — Überwachung: was zu überwachen ist
- **ID-55** — Prüfliste vor der Freischaltung