Ordivis

TEIL V · ADMINISTRATION

Administrationshandbuch

Ersteinrichtung, Einstellungen, Jobs, Branding, Import/Export

Produktversion 2026.08.19 · Handbuch-Revision 1.0

Stand 19.08.2026 · Plattform .NET 10 / Windows Server

ID-40 Administration

Administrationshandbuch

Dieses Dokument bündelt die zentralen Verwaltungsaufgaben. Spezialthemen sind ausgelagert: Benutzer & Rollen (ID-41), Mandantenfähigkeit (ID-42), Produktlizenzierung (ID-43), Installation/Serverbetrieb (ID-50). Alle Einstellungen erreichen Sie über **Einstellungen** im Client (Recht `system.config.manage`).

INHALT

1 Ersteinrichtung (First-Run)

> 💻 **Nur im Client:** erscheint beim Erststart von selbst · 🔑 kein Recht nötig – es gibt noch keine Konten — Das Portal setzt einen eingerichteten Mandanten voraus und kann den Erststart nicht übernehmen.

🔒 **Pflichtreihenfolge** — der erste Administrator gehört zum Hauptmandanten. Ohne Schritt 1 gibt es niemanden, dem Schritt 2 ihn zuordnen könnte; der Assistent lässt die Schritte deshalb nicht tauschen.

1. Legen Sie den **Hauptmandanten** an (Name, Standort/Zentrale).
2. Legen Sie den **ersten Administrator** an – hier **mit** Kennwort. Das ist die einzige Stelle, an der ein Konto direkt mit Kennwort entsteht.
3. Bestätigen Sie. Ordivis Platform erzeugt automatisch die **Stammdaten**: Standardrollen, Standard-SLA-Profile und Standard-Kategorien.

✅ **Ergebnis:** Sie können sich mit dem angelegten Konto anmelden und danach weitere Mandanten, Benutzer und Konfigurationen ergänzen.

> **Achtung:** Das Kennwort des ersten Administrators ist der einzige Zugang zum frisch eingerichteten System. Es gibt an dieser Stelle **keinen** Wiederherstellungsweg – notieren Sie es sicher, bevor Sie fortfahren.

2 Verzeichnisanbindung (AD/LDAP, SSO)

- **AD/LDAP-Sync** konfigurieren (Server, Bind-DN, Basis-DN, Intervall) und **testen**.
- Der Sync ist **immer read-only** – Ordivis Platform schreibt nie ins AD.
- **SSO/SAML** koppelt externe Identitäten an Ordivis-Konten.

Abbildung 1 zeigt die Konfigurationsseite samt Verbindungstest.

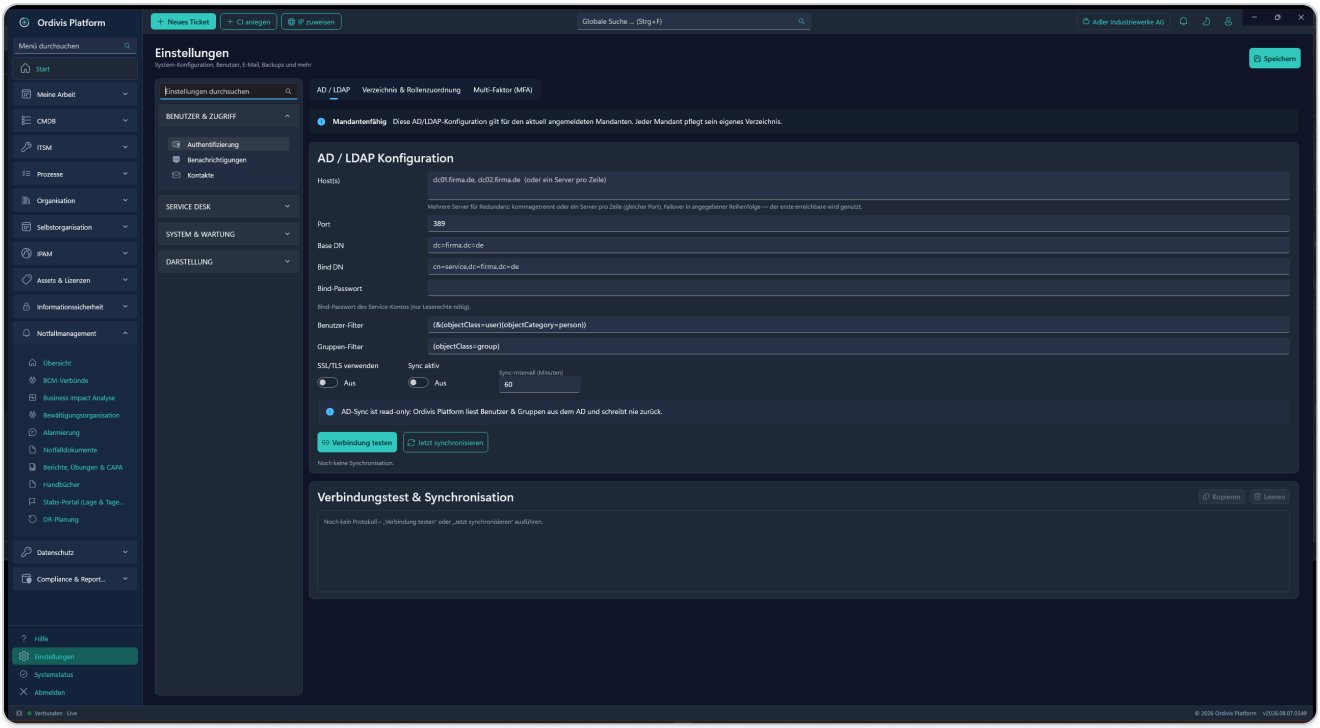

Abb. 1 – AD/LDAP-Konfiguration (mandantenspezifisch): Host, Base-/Bind-DN, Filter, Sync-Intervall und Verbindungstest – der Sync ist read-only.

Die Zuweisung von Rollen zu synchronisierten Benutzern erfolgt in der Benutzerverwaltung (ID-41).

3 Service-Desk-Konfiguration

Siehe ID-20, Abschnitt „Konfiguration“: SLA-Profile, Geschäftszeiten/Kalender, Kategorien + CMDB-Vorfilter, Auto-Routing, Freigabestufen, Wartungsfenster.

4 E-Mail-Integration

Siehe ID-31: SMTP-Versand, IMAP-Eingang, OAuth2, Benachrichtigungen, Präferenzen.

4.1 Fehlerberichte an den Hersteller-Support

Anwender können aus dem Client heraus einen Fehlerbericht senden (Käfer-Symbol in der Statusleiste, siehe ID-11). Für Sie als Administrator sind vier Dinge wichtig:

- **Der Versand läuft über das Outbound-Postfach Ihres Mandanten**, nicht über ein Postfach des Herstellers. Ohne eingerichteten E-Mail-Versand (ID-31) bleibt nur der lokale Weg: Der Client legt den Bericht dann als ZIP unter *Dokumente* ab und öffnet einen E-Mail-Entwurf.
- **Auf dem Server wird nichts gespeichert.** Der Bericht wandert in die Versand-Warteschlange und mit der Zustellung wieder heraus – es gibt keine Fehlerbericht-Tabelle.
- **Die Empfängeradresse ist konfigurierbar.** Voreingestellt ist `support@ordivis.eu`; über `Support:ReportAddress` in der `appsettings.json` leiten Sie die Berichte auf ein eigenes Postfach um, etwa wenn Ihr First Level sie zuerst sichten soll.

- **Der Bericht geht als von Hand geschriebene Mail hinaus** (`Auto-Submitted: no` nach RFC 3834) – anders als Benachrichtigungen und Eingangsbestätigungen, die als maschinell erzeugt gekennzeichnet sind. Nur so erzeugt ein Ticketsystem auf der Empfängerseite daraus einen Vorgang: Sein Schleifenschutz verwirft Maschinenmails, damit eine Abwesenheitsnotiz kein Ticket auslöst. Das ist besonders wichtig, wenn Sie mit `Support:ReportAddress` auf ein **eigenes Ticket-Postfach** umleiten. Als Antwortadresse trägt der Bericht die **meldende Person** – nicht das Postfach, über das er hinausging. Zur Zuordnung enthält er außerdem **Lizenznehmer** und **Mandantenname** (nicht dessen ID).

Der Endpunkt steht **jedem angemeldeten Benutzer** offen und verlangt bewusst kein eigenes Recht: Wer auf einen Fehler läuft, ist häufig der Anwender mit den wenigsten Rechten. Was mitgeschickt wird, entscheidet der Anwender im Dialog Punkt für Punkt; ohne Häkchen wird die betreffende Quelle gar nicht erst gelesen.

4.2 Push-Benachrichtigungen an die Android-App

Push ist Komfort, kein Zustellweg. Fällt die Zustellung aus, geht **keine Meldung verloren**: Sie bleibt in der Warteschlange, wird bis zu sechsmal mit wachsendem Abstand erneut versucht, und sie steht ohnehin in der Anwendung – die App zeigt sie beim nächsten Öffnen. Ein Ausfall des Zustellwegs ist damit ein Komfortverlust, kein Datenverlust.

Zwei Betriebsarten stehen zur Wahl (`Push:Mode` in der `appsettings.json`):

Wert	Bedeutung
`Relay` (Vorgabe)	Die Zustellung läuft über den Vermittler bei Grams IT. Auf diesem Server liegt **kein** Google-Schlüssel. Ausgewiesen wird sich mit dem Aktivierungs-Lease, das die Online-Aktivierung ohnehin führt.
`Direct`	Der Server sendet selbst an Firebase Cloud Messaging und braucht dafür einen eigenen Dienstkonto-Schlüssel im Abschnitt `Fcm` (DPAPI-geschützt). Nur sinnvoll mit einem **eigenen** Firebase-Projekt – das setzt einen eigenen App-Build voraus.

⚠️ **Ohne Online-Aktivierung gibt es kein Push.** Im Vermittler-Betrieb ist das Aktivierungs-Lease der Ausweis; eine Anlage ohne Verbindung ins Internet hat keines. Das ist keine Einschränkung des Vermittlers: Ohne Weg ins Internet erreicht auch der unmittelbare Versand Googles Zustelldienst nicht.

Nach einem Neustart steht das Lease erst mit dem ersten Lizenz-Heartbeat (Vorgabe: zehn Minuten). Meldungen aus diesem Fenster werden deshalb als **vorübergehend gescheitert** geführt und wiederholt – nicht verworfen.

Welche Daten dabei wohin fließen und was protokolliert wird, steht öffentlich unter https://ordivis.eu/unterauftragsverarbeiter.html.

5 Geplante Aufgaben (Job-Scheduler)

> 💻 **Nur im Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Geplante Aufgaben* · 🔑 `system.config.manage` — Das Portal zeigt den Betriebszustand unter *Compliance & Reporting* ▸ *Betriebssicht*, steuert die Aufgaben aber nicht.

Der zentrale **Managed Job Scheduler** verwaltet wiederkehrende Aufgaben:

1. Öffnen Sie *Einstellungen* ▸ *Geplante Aufgaben*.
2. Sehen Sie die Liste durch – je Job stehen dort *Zeitplan*, *Status* und der letzte Lauf.
3. Schalten Sie einen Job über den Schalter in seiner Zeile **aktiv** oder **inaktiv**.
4. Lösen Sie einen Job bei Bedarf über *Jetzt ausführen* **manuell** aus.

✅ **Ergebnis:** Der Job läuft zum eingestellten Zeitpunkt. Nach einem manuellen Lauf steht das Ergebnis in der Spalte für den letzten Lauf.

> **Tipp:** Ein Job, der seit Tagen nicht mehr gelaufen ist, fällt in dieser Liste sofort auf – und sonst nirgends. Sehen Sie hier turnusmäßig nach.

Abbildung 2 zeigt die Liste der Aufgaben.

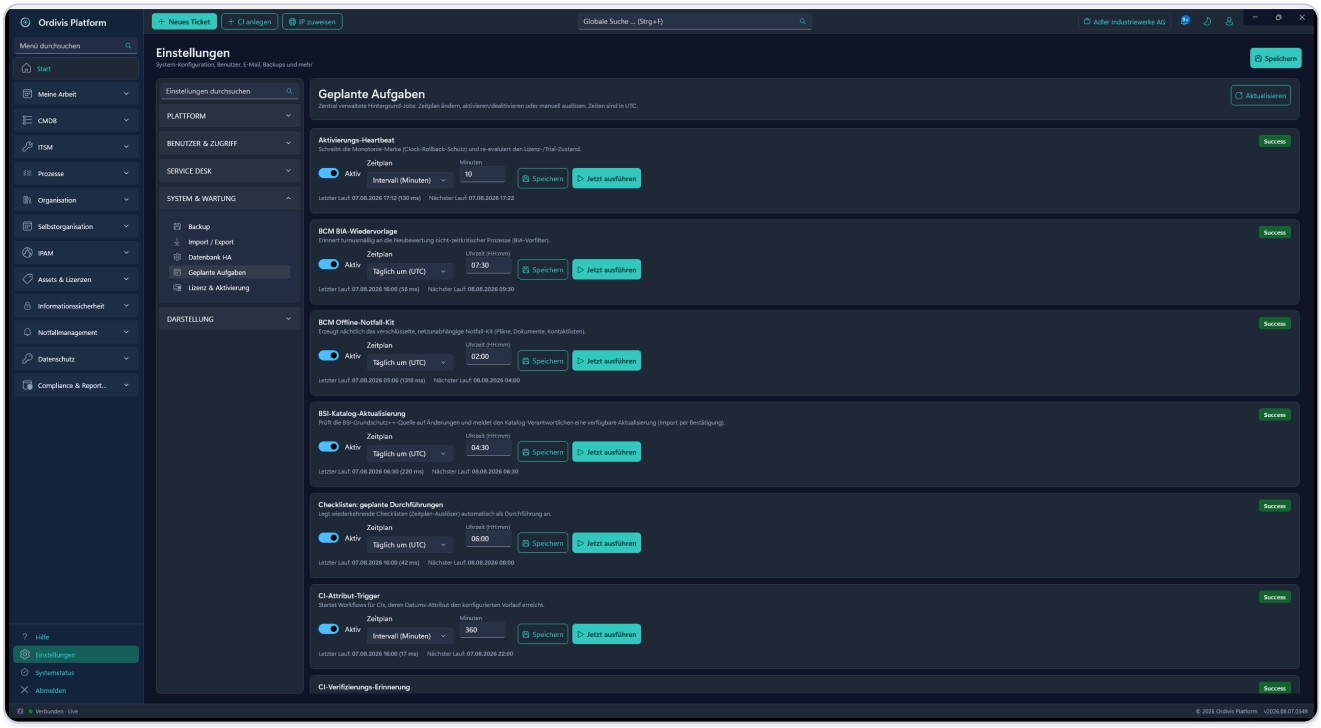

Abb. 2 – Geplante Aufgaben: zentral verwaltete Hintergrund-Jobs mit Zeitplan, letztem/nächstem Lauf und Status.

> **Hinweis:** Einige Integrationen (IMAP, LDAP, Backup, geplante Berichte) bringen eigene Zeitplan-Konfiguration mit und erscheinen zusätzlich in ihrem jeweiligen Bereich.

5.1 Aufgaben und Mandanten

Die Liste ist **installationsweit**: Jede Aufgabe steht genau einmal darin, und Zeitplan, Status und letzter Lauf gelten für die gesamte Installation – nicht je Mandant. Was eine Aufgabe *bearbeitet*, hängt dagegen von ihrer Art ab:

- **Mandantenbezogene Aufgaben** (der weit überwiegende Teil: SLA-Fristen, Wiedervorlagen, Vertrags- und Lizenzerinnerungen, Postfachabruf, Mailversand, Kennzahlen) laufen bei jedem Lauf **nacheinander für jeden aktiven Mandanten** und arbeiten dabei jeweils in dessen eigener Datenbank.
- **Plattformaufgaben** (Update-Prüfung, Lizenz-Heartbeat, Plattformüberwachung) betreffen die Installation als Ganzes und laufen genau einmal.

Daraus folgen zwei Dinge für den Betrieb: Der angezeigte Status **Erfolgreich** bedeutet, dass der Durchgang über *alle* Mandanten fehlerfrei war; und die Laufzeit einer mandantenbezogenen Aufgabe wächst mit der Zahl der Mandanten. Läuft eine Aufgabe noch, wird sie beim nächsten fälligen Termin nicht ein zweites Mal gestartet, sondern übersprungen – bei sehr kurzen Intervallen und vielen Mandanten ist deshalb ein größerer Abstand die bessere Einstellung.

6 CI-Klassen-Verwaltung

Abbildung 3 zeigt die konfigurierten Klassen mit der Spalte *Felder*.

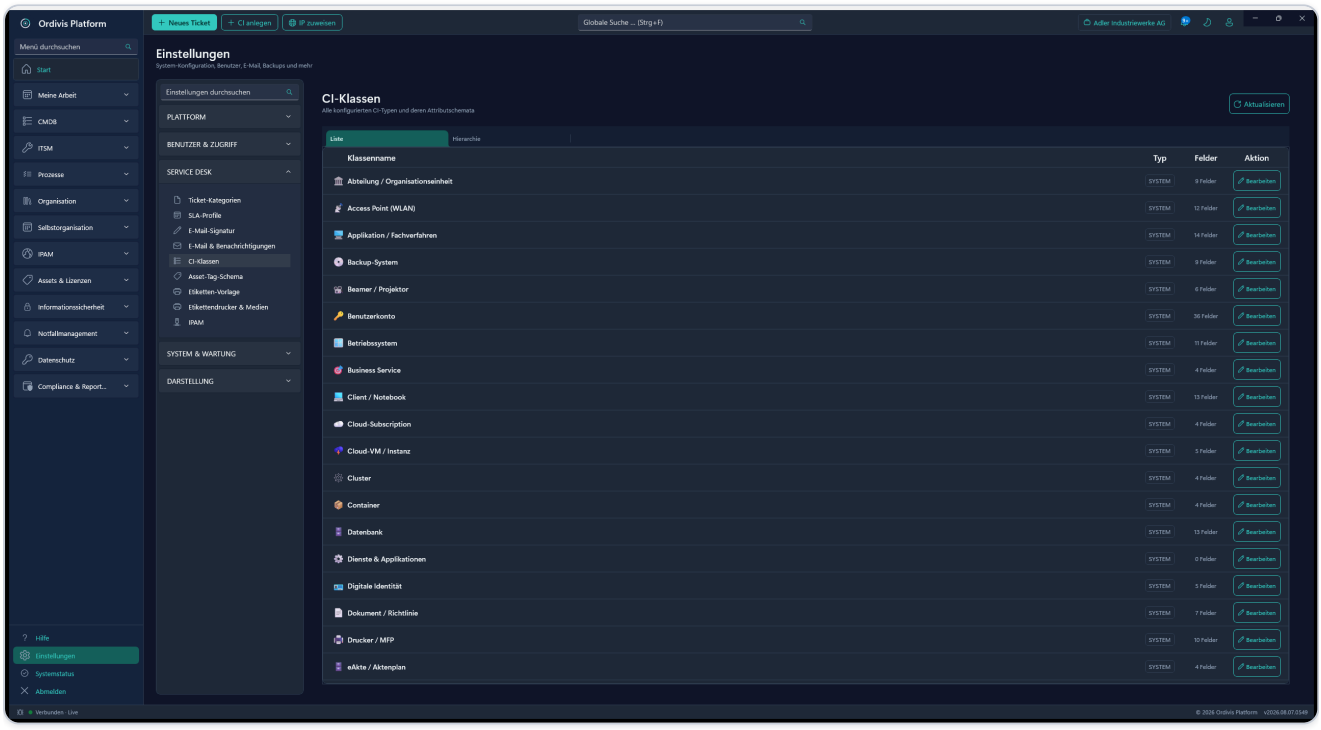

Abb. 3 – CI-Klassen: alle konfigurierten CI-Typen mit Attributschema (Spalte „Felder").

Unter **CI-Klassen** definieren Sie Klassen und Attribute der CMDB (ID-21). Neue Attribute werden zentral angelegt und erscheinen dann automatisch im CI-Formular.

Daneben stehen zwei Einstellungsseiten für die Kennzeichnung der Geräte (Einzelheiten in ID-21, Abschnitt „Etiketten und NFC-Tags am Gerät"):

- **Etiketten-Vorlage** – was auf dem Etikett steht (Barcode-Art, Layout, sichtbare Felder, Logo). Genau eine Vorlage je Mandant.
- **Etikettendrucker & Medien** – worauf gedruckt wird: Maße, Auflösung und Sprache des Geräts (PDF-Bogen, Zebra ZPL, Brother, Epson). Davon dürfen mehrere nebeneinander bestehen; höchstens eines ist die Vorgabe. Ein Testdruck lässt sich erzeugen, bevor das Medium gespeichert wird.

7 Erscheinungsbild / Branding

> 💻 **Nur im Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Erscheinungsbild* bzw. ▸ *Branding* · 🔑 kein Recht für das persönliche Erscheinungsbild, `system.config.manage` für das Branding — Das Portal bietet das persönliche Erscheinungsbild im Paletten-Menü an (ID-12), das mandantenweite Branding jedoch nicht.

Ordivis Platform trennt hier bewusst zwei Ebenen:

- **Erscheinungsbild** (pro Benutzer): heller/dunkler Modus und Farbschema (Material Design 3), dazu Barrierefreiheit (hoher Kontrast, Schriftgröße, reduzierte Bewegung, dauerhaft sichtbarer Fokus) und Sprache.
- **Branding** (pro Mandant): **Firmenname und Logo für generierte Dokumente**. Beide erscheinen als Kopf auf allen Reports und PDFs – Report-Exporte sowie ISMS-/BCM-Berichte (Risikoregister, Schutzbedarf, Erklärung zur Anwendbarkeit, Notfallplan u. a.). In der Fußzeile stehen weiterhin die Ordivis-Kennzeichnung („Erstellt durch Ordivis Platform“) und die Seitenzahl. Das Logo (PNG/JPEG, max. 2 MB) wird zentral hinterlegt und mandantenweise verwendet; Ändern erfordert `system.config.manage`.

Abbildung 4 zeigt die Einstellungen dazu.

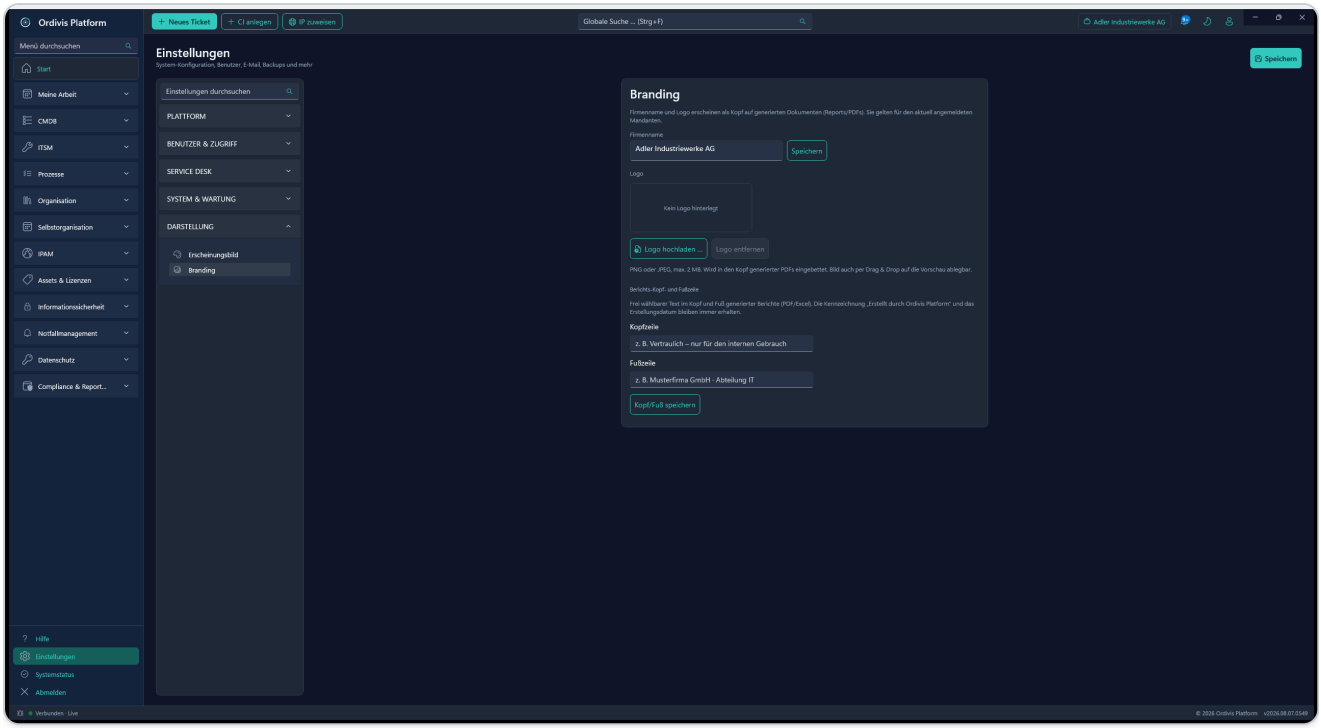

Abb. 4 – Branding (pro Mandant): Firmenname und Logo für den Kopf generierter Reports und PDFs; die Ordivis-Kennzeichnung bleibt in der Fußzeile.

7.1 Sprache der Oberfläche

> 💻 **Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Erscheinungsbild* ▸ *Sprache* · 🌐 **Portal:** Flaggensymbol in der Kopfzeile · 🔑 kein Recht nötig — die Sprache gilt je Benutzer, nicht je Mandant

Die Oberfläche liegt in **Deutsch, Englisch und Französisch** vor. Deutsch ist die Produktsprache; Englisch und Französisch sind vollständig übersetzt.

Im Client stehen Deutsch und Englisch als Schalter bereit, alle weiteren Sprachen in der Liste darunter. Die Namen stehen dort in der jeweiligen Sprache selbst — wer die Oberfläche gerade nicht lesen kann, findet *Français*, aber nicht *Französisch*.

Was Sie wissen sollten:

- Die Umschaltung wirkt im Client **nach einem Neustart des Programms**. Das Portal lädt die Seite selbst neu und ist sofort umgestellt.
- Ist eine Sprache nicht vollständig übersetzt, erscheinen die **fehlenden Texte auf Deutsch**. Es bleibt nichts leer und nichts bricht ab.
- Nicht übersetzt werden **Ihre eigenen Daten**: Ticketbetreffs, CI-Namen, Notizen und Dokumente stehen in der Sprache, in der sie erfasst wurden. Die Oberfläche wechselt, der Inhalt nicht.
- **Datums-, Zeit- und Zahlenformate** folgen der gewählten Sprache. Aus dem 15.08.2026 wird im Englischen 8/15/2026 — dieselbe Angabe, andere Schreibweise.
- Weitere Sprachen sind vorbereitet, aber noch nicht übersetzt; sie erscheinen erst in der Auswahl, wenn ihre Übersetzung vorliegt.

8 Import / Export

Tabellen bieten überall **CSV-/PDF-Export**. Für den Weg hinein – den Dateiimport – gilt Folgendes.

8.1 Was sich per Datei importieren lässt

Bereich	Wo der Import läuft	Was entsteht
CMDB – Configuration Items	CI-Liste ▸ *Import*, oder Einstellungen ▸ Import / Export	CIs einer wählbaren CI-Klasse
Organisation – Aufbauorganisation	Organisation ▸ Aufbauorganisation ▸ *Import*	Einheiten, Stellen, Besetzungen, Rollen
Prozesse – Steckbriefe	Prozesse ▸ Prozessdokumentation ▸ *Import*	Prozesse samt Steckbriefangaben

Andere Bereiche – etwa Tickets oder Assets – haben **keinen** Dateiimport. Die Auswahl in den Einstellungen nennt deshalb genau die Bereiche, die einen haben; sie wird vom Server geliefert und kann nicht mit der Wirklichkeit auseinanderlaufen.

8.2 Die Vorlage ist der kürzeste Weg

> 💻 **Nur im Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Import / Export* · 🔑 `cmdb.ci.update` — Das Portal hat keinen Dateiimport.

🔒 **Pflichtreihenfolge** — die Vorlage wird **je Bereich und CI-Klasse erzeugt**. Wer sie vor der Auswahl herunterlädt, bekommt die Spalten eines anderen Bereichs.

1. Öffnen Sie *Einstellungen* ▸ *Import / Export*.
2. Wählen Sie den **Bereich** – bei der CMDB zusätzlich die **CI-Klasse**.
3. Laden Sie die **Beispieldatei** als CSV oder XLSX herunter.
4. Überschreiben Sie die Beispielzeile mit Ihren Daten.
5. Laden Sie die ausgefüllte Datei hoch und prüfen Sie die **Vorschau**, bevor Sie übernehmen.

✅ **Ergebnis:** Der Import läuft gegen genau die Felder, die der Bereich kennt – ohne dass Sie die Kopfzeile erraten mussten.

Die Beispieldatei enthält:

- die **vollständige Kopfzeile** – alle möglichen Felder des gewählten Bereichs,
- eine **Beispielzeile** mit gültigen Werten, die Sie überschreiben,
- bei XLSX ein zweites Blatt **Feldbeschreibung** mit Typ, Pflichtkennzeichen, Bedeutung und den zulässigen Werten je Spalte.

Die Vorlage wird **erzeugt, nicht gepflegt**: Ihre Spalten kommen aus derselben Quelle, aus der der Import liest. Legen Sie ein neues Attribut an einer CI-Klasse an, steht es ohne weiteres Zutun in der Vorlage. Dieselbe Übersicht zeigt die Schaltfläche **Aufbau der Datei anzeigen** direkt in der Anwendung.

> **Read-only-Felder fehlen bewusst.** Werte, die Ordivis Platform selbst berechnet (etwa aus einer aktiven Asset-Zuweisung), stehen nicht in der Vorlage – ein Wert in der Datei wäre eine Behauptung, die beim nächsten Neuberechnen verschwindet.

8.3 Regeln für die Datei

- **Kopfzeile:** genau die Spaltennamen der Vorlage. Reihenfolge und Groß-/Kleinschreibung sind egal. Nicht benötigte Spalten dürfen leer bleiben oder entfallen – außer den Pflichtspalten.
- **Trennzeichen:** Semikolon oder Komma; Ordivis Platform erkennt es an der Kopfzeile. Felder mit Trennzeichen gehören in Anführungszeichen.
- **Zeichensatz:** UTF-8. Die heruntergeladene CSV-Vorlage trägt eine Byte-Reihenfolge-Marke, damit Excel Umlaute im deutschen Gebietsschema richtig öffnet.
- **Datum:** `2026-12-31` oder `31.12.2026`.
- **Zahlen:** `1234.56` oder `1234,56`, **ohne Tausendertrennzeichen**. `1.234` wird als 1,234 gelesen, nicht als 1234.
- **Unbekannte Spalten** werden gemeldet, nicht stillschweigend übergangen – ein Tippfehler in der Kopfzeile fällt damit auf, statt eine ganze Spalte verschwinden zu lassen.

8.4 Erst der Probelauf, dann der Bestand

Jeder Import beginnt mit einem **Probelauf**: Er prüft jede Zeile, nennt die Zahl der Objekte, die entstehen würden, und listet die Beanstandungen mit Zeilennummer und Feld. Erst nach Ihrer ausdrücklichen Zustimmung wird geschrieben. Der Knopf zum Zustimmen ist bewusst **nicht** vorbelegt.

Bei CIs erkennt der Import Dubletten an der **Seriennummer**: Ist sie bereits vergeben, wird die Zeile übersprungen statt ein zweites Objekt anzulegen. Kommt dieselbe Seriennummer zweimal in *einer* Datei vor, wird die zweite Zeile beanstandet.

Für den Import gilt ein eigenes Recht (`cmdb.import` bei CIs, `gvp.manage` bzw. `process.manage` bei Organisation und Prozessen) – Leserechte allein genügen nicht.

9 Updates

Ordivis Platform aktualisiert sich als **Gesamtprodukt** (API, Web-Portal, WinUI-Client, Wandschirm) über **signaturgeprüfte Delta-Pakete** – übertragen werden nur die tatsächlich geänderten Dateien, nicht das vollständige Setup. Zentrale Anlaufstelle ist **Einstellungen ▸ Updates** im Client (Administratoren: Prüfen/Anzeigen mit `activation.manage`, Auslösen mit `system.config.manage`).

Dieses Kapitel beschreibt das Update-System aus Sicht des Administrators und – im Mittelpunkt – die **Mechanismen, die ein Update sicher und nachprüfbar machen** (Abschnitt 9.3). Die serverseitigen Betriebsdetails (Endpunkte, Verzeichnisse, Fehlerbehebung) stehen im Betriebshandbuch (ID-50).

9.1 Erkennung

Die API prüft serverseitig gegen die Website-`version.json` auf neue Versionen und informiert Administratoren (In-App / E-Mail / Portal), dedupliziert je Version (Ereignis `system.update.available`).

Die Prüfung läuft per Zeitplan-Job und lässt sich über **„Nach Updates suchen"** manuell anstoßen.

9.2 Ablauf eines Update-Laufs

Vom Auslösen bis zum geprüften Ergebnis durchläuft jedes Update dieselben Schritte:

1. **Manifest prüfen.** Die API lädt das signierte Release-Manifest der Zielversion und prüft **Signatur** und **Versionsbindung**. Schlägt das fehl, endet der Lauf sofort – es wird nichts verändert.
2. **Datenbank sichern.** Vor jedem Eingriff wird das tägliche DB-Backup angestoßen (Sicherheitsnetz, u. a. für Datenbank-Migrationen).
3. **Delta laden (Staging).** Das Produkt vergleicht die installierten Datei-Hashes mit dem Manifest und lädt **nur die abweichenden** Dateien – jede **einzeln SHA-256-geprüft** – in ein Staging-Verzeichnis. Die laufende Installation bleibt dabei unberührt (**nicht-destruktiv**).
4. **An den eigenständigen Updater übergeben.** Die API stellt einen **Gesamtplan** über alle betroffenen Komponenten zusammen und startet den **entkoppelten Updater**. Dieser läuft eigenständig weiter, auch wenn API oder Client währenddessen beendet werden.
5. **Komponenten in Reihenfolge übernehmen.** Der Updater arbeitet den Plan ab – **Dienste zuerst** (Portal, dann API), den **Client zuletzt**. Je Komponente: Dienst **sauber stoppen** → Dateien mit **Snapshot** tauschen → Dienst **starten** → **Health-Check**. Bei Fehler: **Rollback** auf die Vorversion.
6. **Client zuletzt neu starten.** Wurde das Update manuell am Client ausgelöst, startet der Updater den Client **erst dann** neu, wenn die API wieder gesund ist. Ein reines Server-/Auto-Update lässt einen bewusst geschlossenen Client geschlossen.
7. **Geprüftes Ergebnis ablegen.** Der Updater schreibt das Ergebnis je Komponente nach `update-status.json`; der Client zeigt es beim nächsten Start an (Abschnitt 9.5).

> **Tipp:** Weil die Ausführung entkoppelt ist, meldet das Fortschrittsfenster nach dem Übergeben „**an den Updater übergeben – wird ausgeführt und geprüft**". Das ist bewusst **kein** vorschnelles „fertig"; der endgültige, geprüfte Ausgang erscheint nach dem Neustart unter *Einstellungen* ▸ *Updates*.

9.3 Was das Update sicher und verifizierbar macht

Ein Update greift tief in eine laufende Installation ein. Ordivis Platform behandelt es deshalb als **sicherheitskritische Operation**: Mehrere voneinander unabhängige Mechanismen greifen ineinander, und **keiner verlässt sich darauf, dass ein anderer funktioniert hat**. Sie beantworten drei Fragen.

Kommt das Update wirklich vom Hersteller – und ist es unverändert? (Authentizität & Integrität)

- **Signiertes Release-Manifest.** Zu jeder Version gehört ein `manifest.json`, das je Komponente **jede** Datei mit ihrem **SHA-256-Fingerabdruck** und ihrer Größe auflistet. Das Manifest ist mit **ECDSA / ES256** (Kurve P-256, kompaktes JWS) **signiert**. Der **private** Schlüssel liegt ausschließlich beim Hersteller; der **öffentliche** ist fest in das Produkt eingebaut. Ein Manifest ohne gültige Signatur wird **nie** angewandt – der Lauf beginnt gar nicht erst.
- **Bindung an die Zielversion.** Ein zwar korrekt signiertes, aber für eine *andere* Version ausgestelltes Manifest wird abgelehnt. Ein altes (womöglich verwundbares) oder verwechseltes Paket lässt sich damit nicht unterschieben.

- **Jede Datei einzeln geprüft.** Die Dateien liegen **inhaltsadressiert** ab – sie werden über ihren Hash angefordert und nach dem Laden **erneut gehasht** und gegen den signierten Manifest-Wert geprüft. Weicht ein Hash ab, wird die Datei verworfen und der Lauf abgebrochen – **auch dann, wenn der Download- oder Spiegelserver manipuliert wäre**.
- **Zwei Schutzschichten.** Die **Signatur** beweist, dass die *Datei-Liste* echt ist; der **Hash je Datei** beweist, dass *jede geladene Datei* exakt dieser Liste entspricht. Ein manipuliertes Netzwerk oder CDN kann so keinen fremden Code einschleusen.
- **Rechtetrennung.** Nur die Plattform-Ebene darf ein Update **auslösen** (`system.config.manage`); schon das **Anzeigen** des Umfangs erfordert `activation.manage` .

Wird nur ein geprüfter Zustand übernommen – und lässt er sich zurücknehmen? (Verifizierbarkeit & Reversibilität)

- **Snapshot + automatischer Rollback je Komponente.** Vor dem Dateitausch wird ein **Snapshot** der Installation angelegt. Scheitert *irgendein* Schritt – Kopieren, Dienststart oder Health-Prüfung –, wird der Snapshot **zurückgespielt** und die **Vorversion** gestartet. Einen „halb aktualisierten" Stand gibt es nicht.
- **Erfolg wird gemessen, nicht angenommen.** Eine Dienst-Komponente gilt erst dann als aktualisiert, wenn ihr Dienst wieder **läuft** *und* ihr **Health-Endpunkt** „gesund" meldet. Andernfalls: Rollback.
- **Geschützter Laufzeit- und Konfigurationsstand.** `appsettings*` , Schlüsseldateien (`*.key`) und der `data/` -Ordner (u. a. der Schlüssel des Mail-Tresors) werden **nie** angefasst – weder beim Entfernen überzähliger Dateien noch beim Rollback. Das verhindert die frühere Fehlerklasse, bei der ein Update Konfiguration oder gespeicherte Geheimnisse verlieren konnte.
- **Entkoppelte Ausführung.** Die Übernahme läuft in einem **eigenständigen Updater-Prozess**, der den Stopp von API und Client **überlebt**. Weil kein laufender Prozess seine eigenen Dateien tauscht, entstehen **keine Datei-Sperren** und damit **keine Teil-Updates**.
- **Feste Reihenfolge, Client zuletzt.** Erst die Dienste (Portal, dann API), dann der Client – dieser kommt **erst hoch, wenn die API aktualisiert und gesund ist**.
- **Sicherheitsnetz vor dem Eingriff.** Vor dem Laden der Dateien wird das **Datenbank-Backup** angestoßen; das Staging selbst ist **nicht-destruktiv**.

Kann man hinterher zweifelsfrei sehen, was passiert ist? (Nachvollziehbarkeit)

- **Geprüftes Ergebnis je Komponente.** Der Updater schreibt Fortschritt und geprüftes Endergebnis in `update-status.json` . Der Client zeigt nach dem Neustart **genau dieses** Ergebnis – Erfolg, Rollback oder Fehler je Komponente (Abschnitt 9.5).
- **Revisionssicheres Audit-Ereignis.** Jeder Lauf erzeugt `system.update.performed` mit Zielversion und Ergebnis (Abschnitt 10).
- **Detailprotokoll je Lauf.** Host- und Updater-Protokolle unter `%ProgramData%\InfraDesk\update-logs\` halten jeden Schritt mit Zeitstempel fest; im Update-Fenster ist der Text per Klick kopierbar (Abschnitt 9.6).

Kurzüberblick der Mechanismen

Mechanismus	Sichert …	Woran erkennbar / prüfbar
Signiertes Manifest (ES256)	Echtheit – kommt vom Hersteller	ungültige Signatur → Lauf startet nicht
Versionsbindung des Manifests	kein Unterschieben fremder/alter Pakete	Fremdversion → „passt nicht zur angeforderten"
SHA-256 je Datei (inhaltsadressiert)	Unversehrtheit jeder Datei – auch bei manipuliertem Server	Hash-Abweichung → Datei verworfen, Abbruch
Snapshot + Rollback	kein halb aktualisierter Stand	Fehler → Vorversion läuft wieder
Health-Check	nur ein lauffähiger Zustand zählt als Erfolg	„gesund"-Prüfung nach Start, sonst Rollback
Geschützter Laufzeitstand	Erhalt von Konfiguration/Geheimnissen	`appsettings*` , `*.key` , `data/` bleiben unberührt
Entkoppelter Updater	keine Datei-Sperren, keine Teil-Updates	Ausführung überlebt Stopp von API/Client
Pre-Update-DB-Backup	Rückfallpunkt bei Datenfehlern	Backup vor dem Lauf angestoßen
`update-status.json`	geprüftes Ergebnis statt Vermutung	Client zeigt Ausgang je Komponente
Audit + Update-Protokoll	lückenlose Nachvollziehbarkeit	`system.update.performed` , `update-logs\`

Wichtig: Diese Prüfungen sind fest eingebaut und lassen sich nicht abschalten. Erst wenn die **Signatur und alle** Datei-Hashes stimmen, wird überhaupt etwas übernommen; andernfalls bleibt die vorhandene Installation unverändert.

9.4 Update-Center

💻 **Nur im Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Updates* · 🔑 `system.config.manage` — Das Portal wird vom Server-Update mit aktualisiert, hat aber keine eigene Update-Seite.

Administratoren finden hier alles an einer Stelle:

- **Auto-Update-Policy in drei Stufen**: **Aus** / **Nur benachrichtigen** (erkennen + melden) / **Vollautomatisch** – bei Vollautomatik wendet der Server das geprüfte Delta selbsttätig im konfigurierten **Wartungsfenster** (Wochentage + Uhrzeit) an (Health + Rollback wie manuell). Die Policy wird gespeichert und steuert Server-Verhalten und Anzeige.
- **Integriertes Changelog**: die Website führt neben `version.json` eine fortgeschriebene `changelog.json`; das aufbereitete Änderungsprotokoll erscheint direkt im Client – kein Wechsel zur Website nötig. Einträge neuer als die installierte Version sind mit **„Neu in dieser Version"** markiert.
- **Komponentenansicht**: je Komponente (API / Portal / Client / Wallboard) installierte Version, verfügbare Zielversion und ob sie vom Update **betroffen** ist (aus dem signaturgeprüften Delta-Plan, inkl.

Dateizahl/MB).

- **Manuelles Update**: eigener Ein-Klick-Weg „Jetzt manuell aktualisieren" – prüft sofort und wendet ein gültig signiertes Delta direkt an, unabhängig von der eingestellten Auto-Update-Policy.

> **Diese Seite zeigt zwei Rechner.** Versionsinfo und Komponentenansicht beschreiben den **Server** – auch die Zeile „Client", die für die auf dem Server abgelegten Client-Dateien steht. Die Version des Arbeitsplatzes, an dem Sie gerade sitzen, steht in der **Statusleiste unten rechts**. Beide können auseinanderlaufen: Ein auf einem eigenen PC installierter Arbeitsplatz-Client wird vom Server-Update nicht erfasst.

9.4.1 Arbeitsplatz-Client aktualisieren

Liegt der Arbeitsplatz hinter dem Server, meldet die Statusanzeige des Update-Centers **„Server aktuell – dieser Arbeitsplatz veraltet"** und nennt beide Versionen. **„Jetzt aktualisieren"** aktualisiert dann **diesen Arbeitsplatz** über den mitgelieferten Aktualisierungsdienst: Der Client wird dafür beendet, die Dateien werden nach Signatur- und Prüfsummenkontrolle getauscht und der Client neu gestartet; bei einem Fehlschlag kehrt er zur Vorversion zurück. Die Dateien kommen über den eigenen Server – der Arbeitsplatz braucht **keinen Internetzugang**.

Derselbe Weg steht im Hinweisbanner am oberen Rand zur Verfügung. Weil die Installation üblicherweise unter *Programme* liegt, fragt Windows nach Administratorrechten; wird das abgelehnt, meldet Ordivis Platform das ausdrücklich. Fehlt der Aktualisierungsdienst – bei Installationen aus der Zeit vor 2026.07.21.2006 – wird stattdessen auf die **Setup-Datei** aus dem Download-Bereich verwiesen; danach greift das Selbstupdate.

Abbildung 5 zeigt das Update-Center mit einer verfügbaren Version.

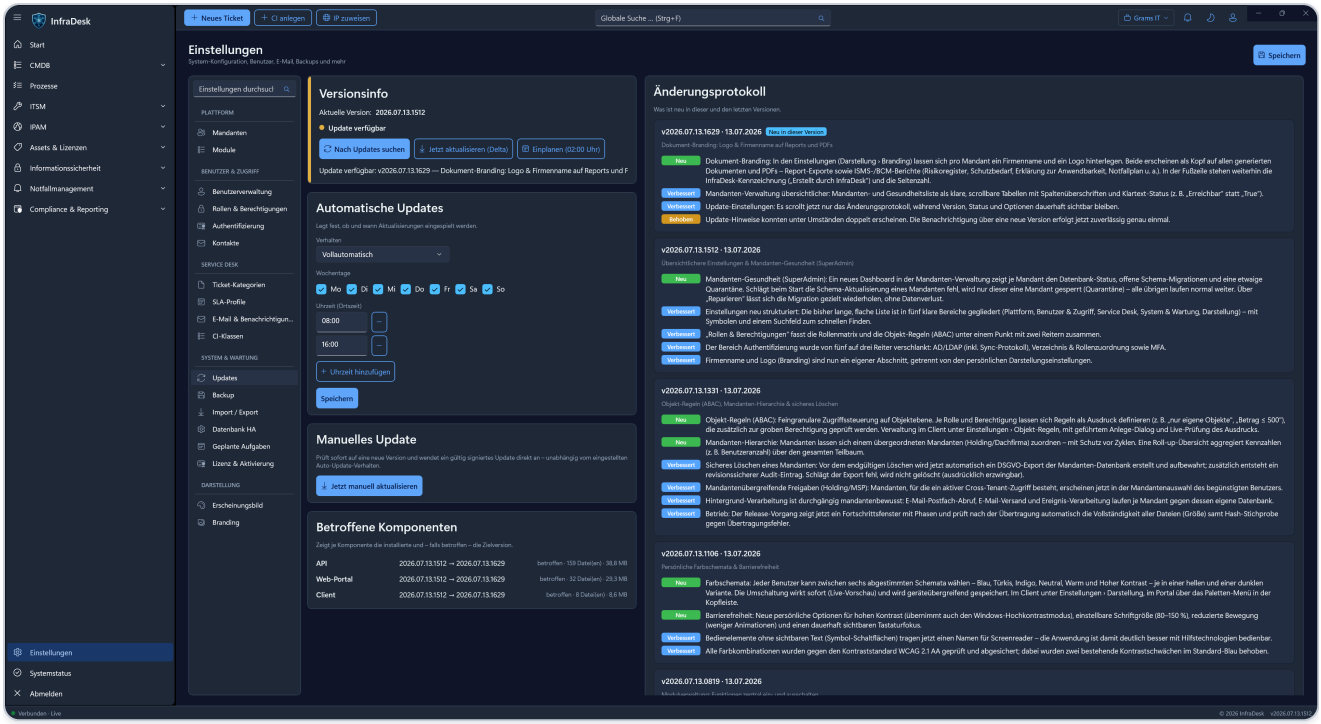

Abb. 5 – Update-Center: eine neuere Version ist verfügbar (2026.07.13.1629). Links Versionsinfo, Auto-Update-Policy mit Wartungsfenster und betroffene Komponenten; rechts das integrierte Änderungsprotokoll.

9.5 Fortschritt und geprüftes Ergebnis

Während des Laufs zeigt ein **Fortschrittsfenster** die Phasen. Weil die eigentliche Übernahme im entkoppelten Updater läuft, endet die Anzeige im Client mit „**an den Updater übergeben**" – das ist bewusst noch **kein** „erfolgreich".

Den Verlauf eines Update-Laufs zeigt Abbildung 6.

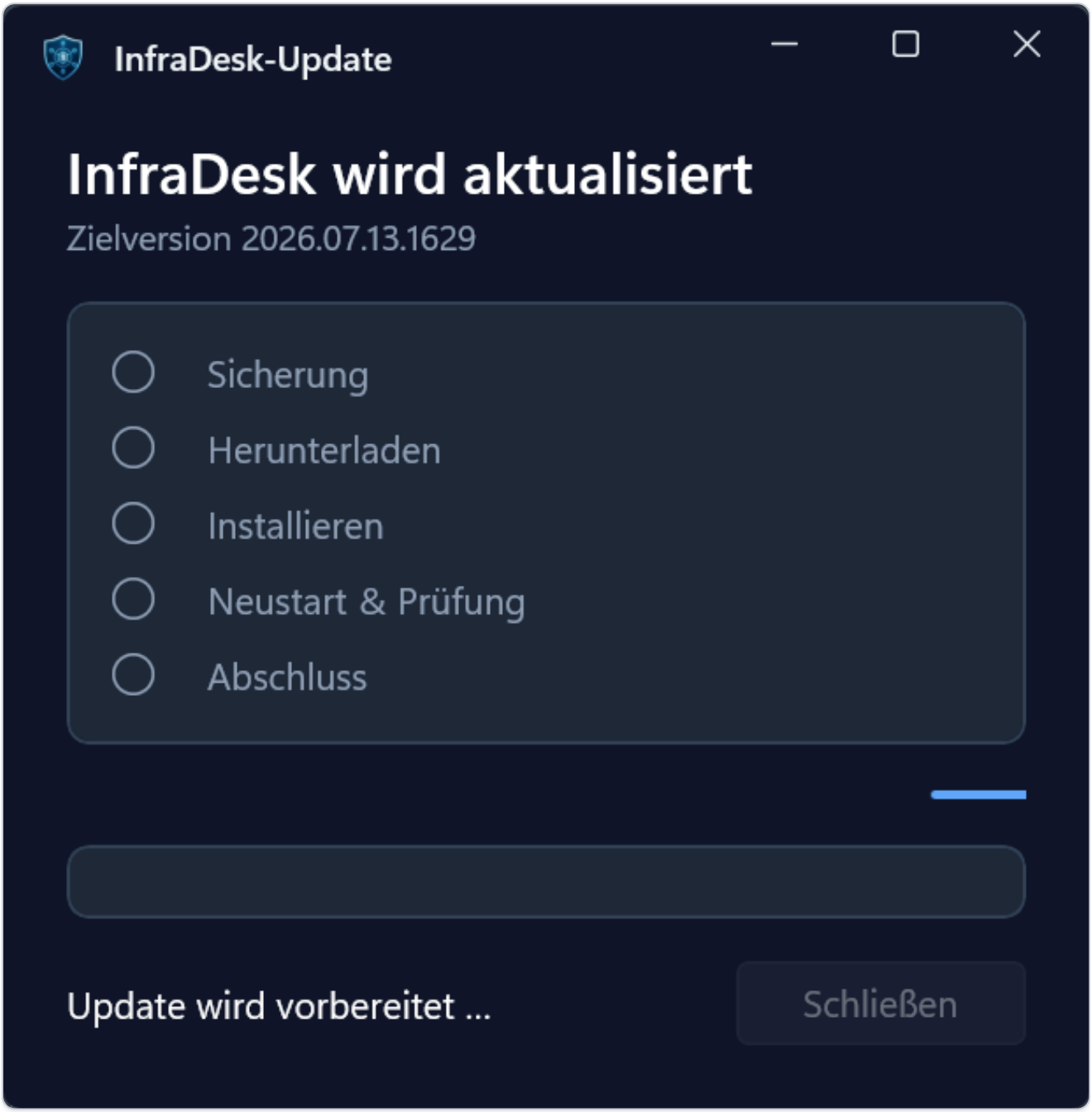

Abb. 6 – Während der Durchführung zeigt ein Fortschrittsfenster die Phasen Sicherung → Herunterladen → Installieren → Neustart & Prüfung → Abschluss (mit automatischem Rollback bei Fehlern).

Das **geprüfte Endergebnis** je Komponente (Erfolg, Rollback oder Fehler) ermittelt der Updater eigenständig und legt es in `update-status.json` ab. Der Client liest diese Datei beim nächsten Start und zeigt das Ergebnis unter **Einstellungen ▸ Updates** an – Sie sehen also den **tatsächlich verifizierten** Ausgang, nicht nur, dass ein Lauf gestartet wurde.

9.6 Update-Protokoll weitergeben

Schlägt ein Update fehl, spielt Ordivis Platform automatisch die Vorversion zurück (Rollback) und zeigt das Protokoll im Update-Fenster. Ein **Klick auf den Protokollbereich** kopiert den gesamten Text in die **Zwischenablage** – Sie können ihn damit in ein eigenes Ticket einfügen oder aus Ihrem Mailprogramm an

den Hersteller-Support senden (`support@ordivis.eu`). Das funktioniert bei fehlgeschlagenen wie bei erfolgreichen Läufen.

> **Ordivis Platform verschickt nichts von selbst.** Es gibt bewusst **keine** Funktion, die Fehlerberichte, Absturzdaten oder Systemmeldungen selbsttätig an den Hersteller übermittelt – keine Telemetrie, kein stiller Absturzbericht, nichts im Hintergrund. Was Ihr Haus verlässt, entscheiden ausschließlich Sie.
>
> Zwei Wege stehen Ihnen dafür offen, und **beide** setzen Ihre Handlung voraus: Sie kopieren das Protokoll und versenden es selbst (Absatz *Klick auf den Protokollbereich* über diesem Hinweis) – oder Sie nutzen den **Fehlerbericht aus dem Client** (Abschnitt 4.1). Dieser übermittelt genau das, was Sie im Dialog Punkt für Punkt bestätigen, und er läuft über das E-Mail-Postfach **Ihrer** Organisation, nicht über eines des Herstellers.

Was im Protokoll steht: die Schritte des Update-Laufs mit Zeitstempel (Sicherung, Herunterladen, Installieren, Dienst-Neustart, Health-Check) sowie die Fehlermeldung, an der der Lauf gescheitert ist – einschließlich Dateipfaden der Installation. Prüfen Sie es vor dem Weitergeben, wenn Ihre Richtlinien das verlangen; Sie sehen im Fenster exakt den Text, den Sie kopieren.

9.7 Rollendes Support-Fenster

Damit Installationen sicher und aktuell bleiben, führt Ordivis Platform ein **rollendes Support-Fenster**. Die Website veröffentlicht in `version.json` eine **Mindestversion** (`minimumVersion`) und kennzeichnet **Sicherheitsreleases** je Eintrag als **verpflichtend** (`mandatory`). Der Client leitet daraus einen gestuften **Support-Status** ab (die Produktversion ist datumsbasiert `JJJJ.MM.TT.HHMM`, das Alter also direkt ablesbar):

- **Unterstützt** – innerhalb des Fensters, kein Hinweis.
- **Alternd** – nähert sich dem Ablauf: **frühzeitige, planbare Warnung**, ein Update einzuplanen.
- **Nicht mehr unterstützt** – unter der Mindestversion **oder** ein Pflicht-(Sicherheits-)Release steht aus: ein **nicht schließbarer Hinweis** mit Direktaktion „Jetzt aktualisieren" fordert das Update bestimmt ein.

Wichtig: Die Durchsetzung erzwingt stets nur das **Update**, sie **blockiert nie den laufenden Betrieb**. Ohne Server-Kontakt (offline / air-gapped) greift kein harter Zustand – eine getrennte Installation wird niemals ausgesperrt. Die Schwelle für „alternd" ist über `Update:SoftFloorDays` (Standard 90 Tage) einstellbar; die harte Untergrenze kommt aus `minimumVersion`.

Kundennutzen: aktuelle Sicherheitspatches, Fehlerbehebungen und Funktionen; ein bekanntes, kleines Versions-Spektrum vereinfacht Support und Migrationspfad – der zulässige Rückstand ist gedeckelt.

9.8 Aufräumen des Delta-Stores (Retention)

Der Publish-Dienst räumt den Delta-Store (`updates/`) **reference-basiert** auf: Er behält die **aktuelle Version**, die **Mindestversion** und die **neuesten Releases** (`Retention:KeepReleases`, Standard 12) und entfernt ältere Versionsordner sowie jeden Blob, den **kein behaltenes Manifest referenziert**. Es wird **nie nach Alter** gelöscht (das neueste Manifest teilt sich viele langlebige Blobs mit älteren Versionen). Kunden werden dadurch **nie ausgesperrt**, weil ein Update immer auf das **vollständige neueste Manifest** zielt und

ein weit zurückliegender Client direkt darauf springt. Das serverseitige Spiegel-Löschen ist bewusst **opt-in** (`Retention:RemotePrune`).

10 Audit-Log

Sicherheitsrelevante Aktionen sind revisionssicher im **Audit-Log** einsehbar (ID-27). Der Zugriff erfordert das Recht `audit.read`.

Was ein Eintrag festhält. Zeitpunkt, handelnde Person (Benutzerkennung und Anmeldename), die ausgeführte Aktion, Art und Kennung des betroffenen Objekts, die IP-Adresse sowie – soweit vorhanden – die geänderten Werte. Zusätzlich wird festgehalten, **worüber** die Änderung kam (Windows-Client, Self-Service-Portal, mobile App oder Schnittstelle) und von **welchem Gerät** (eine je Installation vergebene Kennung ohne Personenbezug). Das ist kein Beiwerk: Eine Inventurbestätigung, die vom Telefon im Serverraum kommt, wiegt bei einer späteren Prüfung anders als dieselbe Buchung vom Schreibtisch, und über die Gerätekennung lassen sich alle Buchungen eines verlorenen Geräts finden. Bei Einträgen aus der Zeit vor Einführung dieser beiden Angaben bleiben sie leer.

Suchen. Filtern lässt sich nach Objektart, einzelnem Objekt, handelnder Person und Zeitraum; die Ausgabe erfolgt seitenweise (höchstens 200 Einträge je Seite).

Mandantengrenze. Die Suche ist immer auf den Mandanten der angemeldeten Person begrenzt. Ein abweichender Mandant lässt sich auch dann nicht abfragen, wenn er in der Anfrage mitgegeben wird.

Ausgeben. Über **Exportieren** wird das Audit-Log mit den *aktuell eingestellten Filtern* als CSV-Datei ausgegeben (Semikolon getrennt, UTF-8 – Excel öffnet sie direkt). Dafür ist das eigene Recht `audit.export` nötig; `audit.read` allein genügt nicht, denn eine ausgegebene Datei verlässt das System und lässt sich nicht zurückholen. Ohne das Recht ist die Schaltfläche nicht sichtbar, und der Abruf wird auch dann abgewiesen, wenn er direkt an die Schnittstelle gerichtet wird.

Die Datei beginnt mit drei Kommentarzeilen: Erzeugungszeitpunkt, abgedeckter Zeitraum und die Angabe, ob der Auszug **vollständig** ist. Aus Rücksicht auf den Arbeitsspeicher des Servers ist die Ausgabe auf 50.000 Einträge begrenzt; wird diese Grenze erreicht, steht das ausdrücklich im Kopf der Datei – grenzen Sie in diesem Fall den Zeitraum ein und geben Sie mehrere Teilauszüge aus.

11 Modulverwaltung (Funktionsmodule ein-/ausschalten)

> 💻 **Client:** *Fußbereich* ▸ *Einstellungen* ▸ *Module* · 🌐 **Portal:** *Verwaltung* ▸ *Mandanten und Module* · 🔑 `tenant.manage`

Nicht jede Organisation braucht alle Funktionen. Der **SuperAdmin** kann einzelne Funktionsmodule zentral aktivieren oder deaktivieren.

🔒 **Pflichtreihenfolge** — der globale Standard wirkt nur dort, wo **kein** Mandanten-Override gesetzt ist. Entfernen Sie deshalb zuerst mit *Global übernehmen* den Override, bevor Sie sich über einen wirkungslosen globalen Schalter wundern.

1. Öffnen Sie die Modulverwaltung.
2. Setzen Sie je Modul den **globalen Standard**.
3. Wechseln Sie bei Bedarf auf einen einzelnen **Mandanten** und setzen Sie dort einen abweichenden Wert.
4. Um einen Override wieder zu entfernen, klicken Sie auf *Global übernehmen*.

✅ **Ergebnis:** Ein deaktiviertes Modul verschwindet aus der Navigation von Client **und** Portal; seine REST-Endpunkte antworten mit **HTTP 404**.

> **Achtung:** Ein abgeschaltetes Modul löscht keine Daten – es blendet sie aus. Beim Wiedereinschalten ist alles unverändert da. Wer Daten wirklich loswerden will, ist hier falsch.

Geltungsbereich. Je Modul lässt sich ein **globaler Standard** setzen; zusätzlich kann pro **Mandant** ein abweichender Wert (Override) hinterlegt werden. Es gilt: *Mandanten-Override sticht globalen Standard, sonst globaler Standard, sonst „aktiv“*. Mit „Global übernehmen“ wird ein Mandanten-Override wieder entfernt.

Wirkung. Ein deaktiviertes Modul verschwindet aus der Navigation von Client **und** Portal; die zugehörigen REST-Endpunkte antworten mit **HTTP 404** (das Modul „existiert“ nach außen nicht). Die Absicherung wirkt serverseitig – nicht nur durch Ausblenden in der Oberfläche.

Pflichtmodule (immer aktiv, nicht abschaltbar): CMDB, Identität & Zugriff, Audit-Log, Benachrichtigungen, Produktaktivierung, Lizenzierung sowie Berichte & Dashboards.

Abschaltbare Module: Service Desk, Wissensdatenbank (eigenständig), IPAM, Discovery, Verträge & Assets, Notfallmanagement (BCM), Wiederanlaufplanung (DR), ISMS (Security), Workflow und Datenschutz (VVT).

> Berechtigung: Die Modulverwaltung ist der Plattform-Ebene vorbehalten und erfordert das Recht `tenant.manage`.

12 Plattform-Überwachung

Das Dashboard **Plattform-Überwachung** zeigt den Zustand der Installation auf einen Blick. Es aktualisiert sich automatisch alle 10 Sekunden, solange die Seite geöffnet ist.

Bereich	Kennzahlen
PostgreSQL	Version und Rolle, aktive Verbindungen gegen Maximum (inkl. Auslastung), Cache-Trefferquote, Commits/Rollbacks, lang laufende Abfragen, Gesamtgröße
Sitzungen	aktive Sitzungen, aktive Benutzer, Anmeldungen der letzten 24 h
Je Mandant	Status, Datenbankgröße, aktive Sitzungen

Worauf zu achten ist: Eine dauerhaft hohe **Verbindungsauslastung** oder eine fallende **Cache-Trefferquote** deuten auf Engpässe hin; **lang laufende Abfragen** sind der übliche erste Hinweis bei Trägheit. Ist ein

Mandant **nicht erreichbar**, wird das ausgewiesen statt still übergangen – die übrigen Kennzahlen bleiben gültig.

„–" heißt „nicht ermittelbar", nicht „null". Kann eine Größe nicht erhoben werden, zeigt das Dashboard einen Strich und liefert die übrigen Kennzahlen weiter – statt die ganze Seite leer zu lassen. Die **Datenbankgrößen** sind dabei ein Sonderfall: Sie erhebt PostgreSQL, indem es je Datenbank deren Verzeichnisbaum durchgeht, was mit der Zahl der Mandanten teurer wird. Sie werden deshalb nur alle paar Minuten neu bestimmt und dazwischen aus einem Zwischenspeicher bedient – die Zahl kann also etwas älter sein als der Rest der Anzeige. Gezählt wird ausschließlich, was zu Ordivis Platform gehört (Haupt- und Mandanten-Datenbanken); fremde Datenbanken auf demselben Server bleiben außen vor.

Berechtigung: `platform.monitor` (Plattform-Ebene).

13 Sicherheitsereignisse im Windows-Ereignisprotokoll

Zusätzlich zum Audit-Log (Abschnitt 10) schreibt Ordivis Platform sicherheitsrelevante Vorgänge in das **Windows-Ereignisprotokoll** unter der eigenen Quelle **„Ordivis Platform"** im Protokoll *Anwendung*. Damit lassen sie sich ohne Zusatzaufwand an ein vorhandenes **SIEM** anbinden, das Windows-Logs bereits einsammelt.

Erfasst werden unter anderem: Anmeldungen und **fehlgeschlagene** Anmeldungen, Kennwortänderungen, Änderungen an Rollen und Rechten, Dienst-, Update- und Lizenzereignisse.

Welche Ereignisse geschrieben werden, konfigurieren Sie auf Plattform-Ebene – so bleibt das Protokoll auf das beschränkt, was Ihr SIEM auswerten soll.

Das Audit-Log wird dadurch nicht ersetzt. Es bleibt die vollständige, revisionssichere Historie in der Anwendung; das Ereignisprotokoll ist die Schnittstelle nach außen.

Berechtigung: `platform.eventlog.manage` (Plattform-Ebene).

13.1 Ereignis-Referenz für Ihr SIEM

Jeder Eintrag trägt eine feste **Ereigniskennung**. Sie ist der Ankerpunkt für Regeln und Auswertungen und ändert sich nicht: Neue Ereignisse bekommen die nächste freie Nummer ihrer Kategorie, bestehende werden nie umnummeriert. Die Nummer folgt dem Schema *Kategorie* × *1000* + *laufende Nummer*; die Kategorie erscheint im Ereignis zusätzlich als Windows-*Kategorie*.

Authentifizierung

ID	Ereignis	Schweregrad	Bedeutung
1001	`LoginSucceeded`	Information	Anmeldung erfolgreich
1002	`LoginFailed`	Warnung	Anmeldung fehlgeschlagen

ID	Ereignis	Schweregrad	Bedeutung
1003	`Logout`	Information	Abmeldung
1004	`AccountLockedOut`	Warnung	Konto wegen zu vieler Fehlversuche gesperrt
1005	`MobileDeviceRevoked`	Warnung	Mobilgerät abgemeldet (Widerruf durch den Benutzer)

Konto

ID	Ereignis	Schweregrad	Bedeutung
2001	`PasswordChanged`	Information	Kennwort geändert
2002	`PasswordReset`	Warnung	Kennwort administrativ zurückgesetzt
2003	`UserCreated`	Information	Benutzerkonto angelegt
2004	`UserDeactivated`	Warnung	Benutzerkonto deaktiviert

Rollen & Rechte (RBAC)

ID	Ereignis	Schweregrad	Bedeutung
3001	`RoleAssignmentChanged`	Warnung	Rollenzuweisung geändert
3002	`PermissionChanged`	Warnung	Berechtigung geändert
3003	`SuperAdminAction`	Warnung	Plattform-(SuperAdmin-)Aktion

Dienst-Lebenszyklus

ID	Ereignis	Schweregrad	Bedeutung
4001	`ServiceStarted`	Information	Dienst gestartet
4002	`ServiceStopped`	Information	Dienst gestoppt
4003	`HealthCritical`	Fehler	Health-Prüfung kritisch

Update

ID	Ereignis	Schweregrad	Bedeutung
5001	`UpdateApplied`	Information	Update angewendet
5002	`UpdateFailed`	Fehler	Update fehlgeschlagen
5003	`UpdateRolledBack`	Warnung	Update zurückgerollt

Lizenz

ID	Ereignis	Schweregrad	Bedeutung
6001	`LicenseActivated`	Information	Lizenz aktiviert
6002	`LicenseExpired`	Warnung	Lizenz abgelaufen
6003	`LicenseTampering`	Fehler	Lizenz-Manipulationsverdacht

Backup

ID	Ereignis	Schweregrad	Bedeutung
7001	`BackupSucceeded`	Information	Backup erfolgreich
7002	`BackupFailed`	Fehler	Backup fehlgeschlagen

System

ID	Ereignis	Schweregrad	Bedeutung
8001	`CriticalError`	Fehler	Kritischer Systemfehler
8002	`MigrationRun`	Information	Datenbank-Migration ausgeführt
8003	`EventLogConfigChanged`	Warnung	Sicherheitsprotokoll-Konfiguration geändert
8004	`PlatformThresholdBreached`	Warnung	Plattform-Schwellwert überschritten

Zu jedem Eintrag gehören strukturierte **Felder** im Ereignistext, u. a. wer die Aktion ausgelöst hat (`AusgeloestVon`, `AusloeserId`), von wo (`QuellIP`), der betroffene Mandant (`MandantId`) und – je nach Ereignis – der betroffene Benutzer (`BetroffenerBenutzer`) oder die geänderte Rolle. Bei `PermissionChanged` stehen zusätzlich `Vorher` und `Nachher` mit den Rechte-Schlüsseln, sodass im Nachhinein nachvollziehbar ist, **was** sich geändert hat – nicht nur, **dass** sich etwas geändert hat.

13.2 Kanal an ein SIEM weiterleiten

Die Ereignisse liegen im Protokoll **Anwendung** unter der Quelle **Ordivis Platform** – also dort, wo Ihre vorhandene Sammlung sie ohnehin abholt. Zwei übliche Wege:

- **Windows Event Forwarding (WEF)** ohne Zusatzsoftware: Auf dem Collector ein Abonnement anlegen und in der Ereignisauswahl (XPath) nach der Quelle filtern:

```
xml <QueryList> <Query Id="0" Path="Application"> <Select Path="Application">*[System[Provider[@Name='Ordivis Platform']]]</Select> </Query> </QueryList>
```

- **SIEM-Agent** (Splunk UF, Elastic Winlogbeat, Wazuh o. ä.): Kanal `Application` einsammeln und auf den Provider `Ordivis Platform` filtern.

> **Umfang zuerst festlegen.** Was überhaupt geschrieben wird, steuern Sie in der Anwendung (Plattform-Ebene): global, je Kategorie und über einen Mindest-Schweregrad. Es ist sinnvoller, dort zu begrenzen, als später im SIEM zu verwerfen – was nicht geschrieben wird, kostet auch keine Lizenz.

13.3 Warum diese Tabelle nicht von Hand gepflegt wird

Die Referenz oben wird aus dem Ereignis-Katalog der Anwendung erzeugt und durch einen Test gegen ihn abgeglichen. Eine abgetippte Tabelle driftet: Ein neues Ereignis wird ergänzt, die Doku nicht – und Ihr SIEM-Regelwerk beruht ab da auf einer Referenz, die nicht mehr stimmt. Weicht die Tabelle vom Katalog ab, schlägt der Test fehl.

14 Identitätsanbieter (SSO/OIDC)

Unter **Einstellungen → Benutzer & Zugriff → Identitätsanbieter (SSO)** werden OIDC-Anbieter je Mandant gepflegt. Sichtbar mit `system.security.manage` – demselben Recht, das die schreibenden Aufrufe verlangen.

Einzutragen sind Autorität (Issuer), Client-Kennung, Rückleitungsadresse und die Geltungsbereiche. Die **E-Mail-Domänen** entscheiden, ob die Anmeldeseite den Anbieter von sich aus vorschlägt: Ohne Eintrag muss er von Hand gewählt werden.

Probe vor dem Scharfschalten. Die Schaltfläche „Probe ausführen" ruft die Angaben des Anbieters ab und zeigt Issuer, Endpunkte, die Zahl der Signaturschlüssel und ob PKCE mit S256 angeboten wird. Fehlt eine Abmeldeadresse oder S256, meldet sie **Warnung** statt Erfolg – beides hat Folgen im Betrieb: Ohne Abmeldeadresse endet ein Abmelden nur in Ordivis, die Sitzung beim Anbieter bleibt bestehen; ohne S256 startet gar keine Anmeldung.

> **Achtung – Aussperren möglich.** Der Schalter „Anmeldung nur über diesen Anbieter" deaktiviert die Kennwortanmeldung. Ist der Anbieter nicht erreichbar, kommt niemand mehr herein. Vor dem Einschalten die Probe erfolgreich durchführen **und** ein Notfallkonto bereithalten, das den Anbieter nicht braucht.

Das **Client-Geheimnis** wird nie angezeigt – die Seite meldet nur, ob eines hinterlegt ist. Ein leeres Feld bedeutet „unverändert" und löscht nichts.

Gruppen auf Rollen abbilden. Der Anspruch mit den Gruppen (meist `groups` oder `roles`) wird eingetragen, darunter die Zuordnungen. Die **Vorschau** rechnet an einem Beispielkonto vor, welche Rollen entstünden – ohne Speichern. Zeigt eine Zuordnung auf eine gelöschte Rolle, gilt die Gruppe als nicht zugeordnet; der Server weist solche Zuordnungen beim Speichern ab.

15 Zugriffsüberprüfung (Access Review)

Unter **Einstellungen → Benutzer & Zugriff → Zugriffsüberprüfung**; Recht `identity.permissions.manage`. Die wiederkehrende Bestätigung, wer welche Rolle hat, ist die Kontrolle, die ISO 27001 unter **A.5.18** verlangt.

Ein Durchlauf entsteht als Entwurf mit Zeitraum. Beim **Starten** sammelt er alle Rollenzuweisungen des Mandanten ein und friert sie ein – ein zweiter Start fügt nichts hinzu, damit nachvollziehbar bleibt, worüber entschieden wurde. Jede Position wird einzeln bestätigt oder entzogen, mit Begründung; Person und Rolle stehen mit Klarnamen da.

Abgeschlossen werden kann erst, wenn keine Position mehr offen ist. Ein abgeschlossener Durchlauf mit offenen Positionen wäre ein Nachweis, der nichts belegt – der Server weist das ab, nicht nur die Oberfläche.

Prüfer und Ersteller stammen ausschließlich aus der Anmeldung und lassen sich nicht setzen. Bei einer Zugriffsüberprüfung ist der Nachweis, *wer* bestätigt hat, der Kern der Kontrolle.

16 Mandantenübergreifende Freigaben

Unter **Einstellungen** → **Plattform** → **Mandanten-Freigaben**; Recht `tenant.grant.manage` (Plattform-Super-Admin). Für Holding-Strukturen und Dienstleistungsverträge: Eine Person erhält zeitlich begrenzten Zugriff auf einen fremden Mandanten. **Die Vorgabe ist aus** – ohne Freigabe bleibt jeder auf seine eigenen Mandanten beschränkt.

Die Rechte werden aus dem Katalog gewählt, nicht getippt: Ein Recht, das es nicht gibt, fiele erst beim Zugriff auf, und dann still. Sie müssen im gewährenden Mandanten existieren.

> **Achtung – ohne Enddatum läuft eine Freigabe nie ab.** Die Liste weist solche Einträge ausdrücklich aus. Für einen Dienstleistungsvertrag gehört das Vertragsende in das Feld „Gültig bis“.

Widerrufene und abgelaufene Freigaben bleiben in der Liste stehen; ein Widerrufsknopf erscheint nur, solange die Freigabe tatsächlich wirkt. Jeder Zugriff über eine Freigabe steht im Audit-Log.

17 Automatisierungsregeln für Vorgänge

Unter **Vorgänge** → **Werkzeuge** → **Automatisierungsregeln**; Recht `ticketing.config.manage` zum Pflegen. Regeln der Form Bedingung → Aktion greifen beim Anlegen eines Vorgangs.

Als Bedingung dienen Vorgangsart, Kategorie, ein Textstück im Betreff und die Domäne des Melders; als Aktion das Setzen von Priorität, Support-Gruppe, Kategorie oder eines Mitlesers. Die Regeln laufen in der eingetragenen Reihenfolge; die erste zutreffende mit **„Kette anhalten“** beendet die Verarbeitung.

> **Nicht mit den Routing-Regeln verwechseln.** Beide Regelwerke laufen beim Anlegen, sind aber getrennt: Die Routing-Regeln bilden ausschließlich Kategorie → Support-Gruppe ab.

Die Liste zeigt Bedingung und Aktion ausgeschrieben. Eine Regel ohne Aktion – und ebenso eine, die nur die Kette anhält – wird als wirkungslos ausgewiesen: Die zweite ändert nichts, verdeckt aber alle nachfolgenden Regeln.

18 Protokollierung, Diagnose und Support-Paket

Unter **Einstellungen** → **System & Wartung** → **Protokolle**; nur Plattform-Ebene. Hier werden Speicherort, Aufbewahrung und Rotation gesetzt sowie die Mindeststufe je Bereich – getrennt nach Technik, Anwendungen und Fachmodulen. **Die Umstellung wirkt ohne Dienst-Neustart.**

Das **Support-Paket** steht bewusst ganz oben auf derselben Seite: Es sammelt Protokolle aller Anwendungen, Update-Protokolle, den Steckbrief der Anlage, eine geschwärzte Konfiguration und einen Auszug des Windows-Ereignisprotokolls in ein AES-256-verschlüsseltes Archiv.

> **Das Kennwort gehört nicht in dieselbe Nachricht wie das Archiv.**

Was aufgenommen und was ausgelassen wurde, steht in der Datei `01-INHALT.txt` im Archiv.

19 Datenschutz: Betroffenenrechte und AVV-Register

Zwei Seiten unter **Datenschutz** in der Hauptnavigation.

Betroffenenrechte (Recht `privacy.manage`) ist das Werkzeug des Datenschutzbeauftragten, keine Selbstbedienung: Über die E-Mail-Adresse werden Konten, Kontakte und Vorgänge einer Person zusammengestellt. Die Ausgabe erfolgt als **JSON** – Art. 20 DSGVO verlangt ein maschinenlesbares Format. Steht zu einer Adresse nichts, ist auch das eine gültige Auskunft nach Art. 15 und der betroffenen Person mitzuteilen.

Die **Löschung nach Art. 17 anonymisiert**, sie löscht nicht: Vorgänge und Nachweise bleiben auswertbar, die Person ist darin nicht mehr erkennbar. Sie ist erst möglich, nachdem die Auskunft erstellt wurde – der Schritt lässt sich nicht rückgängig machen, und wer vorher gesehen hat, was daran hängt, entscheidet anders.

AVV-Register (Rechte wie am Verarbeitungsverzeichnis: `datenschutz.vvt.read` zum Lesen, `.edit` zum Pflegen) führt die Auftragsverarbeitungsverträge nach Art. 28. Abgelaufene stehen oben und farblich abgesetzt – ein abgelaufener Vertrag zu einem aktiven Auftragsverarbeiter ist ein Befund und zählt in der Kachel „AVV abgelaufen“ des Verarbeitungsverzeichnisses mit. Der Dienstleister wird aus der CMDB benannt und dort gepflegt, nicht hier.

20 Eigenes Postfach für den Exchange-Kalender

Keine Administrationsaufgabe, aber hier erwähnt, weil danach gefragt wird: Jeder Benutzer gibt sein Postfach **selbst** frei – im Portal unter **Mein Konto**, im Arbeitsplatz-Client unter **Einstellungen → Mein Postfach (Exchange)**.

Ohne diese Freigabe wird kein Postfach gelesen, auch nicht das eines Administrators. Das Dienstkonto mit `ApplicationImpersonation` könnte technisch jedes Postfach der Organisation öffnen – das ist Exchanges Modell und dort nicht abwählbar; begrenzt wird es im Produkt. Einschalten und Widerruf stehen im Sicherheitsprotokoll, und widerrufene Freigaben bleiben in der Liste: Sie sind der Nachweis darüber, in welchem Zeitraum zugegriffen werden durfte.

21 Wichtige Rechte für Administration

Recht	Erlaubt
`system.config.manage`	Systemeinstellungen verwalten (= „Administrator“)
`system.security.manage`	Sicherheits-/Anmeldedaten verwalten
`identity.users.manage` / `identity.roles.manage`	Benutzer bzw. Rollen verwalten (ID-41)
`tenant.manage`	Mandanten- **und** Modulverwaltung (Plattform-/SuperAdmin)
`platform.monitor`	Plattform-Überwachung ansehen
`platform.eventlog.manage`	Windows-Ereignisprotokoll konfigurieren
`identity.permissions.manage`	Zugriffsüberprüfung durchführen (Kap. 15)
`tenant.grant.manage`	Mandantenübergreifende Freigaben erteilen (Kap. 16)
`ticketing.config.manage`	Automatisierungs- und Routing-Regeln pflegen (Kap. 17)
`privacy.manage`	Betroffenenrechte bearbeiten (Kap. 19)

Verwandte Themen

- **ID-41 – Modul Identität, Benutzer & Rollen** – Benutzer, Rollen und Anmeldeverfahren
- **ID-42 – Modul Mandantenfähigkeit** – Mandanten anlegen und trennen
- **ID-43 – Modul Produktlizenzierung** – Lizenz aktivieren und Limits prüfen
- **ID-31 – Modul Benachrichtigungen & E-Mail** – E-Mail-Versand und Posteingang
- **ID-50 – Betriebshandbuch – Installation & Serverbetrieb** – alles, was am Server selbst passiert