

Änderungsprotokoll

Wer wann was von wo geändert hat - bei Aufgaben, Projekten, Personen, Einstellungen, Geld und Zeit. Einträge lassen sich nicht ändern oder löschen, und eine Hash-Kette macht jede spätere Änderung erkennbar.

Überblick

Das Änderungsprotokoll beantwortet die Fragen eines Sicherheits-Reviews, einer internen Revision oder einer Datenschutzbeauftragten: *Wer hat diesem Benutzer die Finance-Rolle gegeben? Wer hat das Fälligkeitsdatum dieser Aufgabe geändert, und von wo? Wer hat die Rechnung heruntergeladen? Wann wurde diese Einstellung ausgeschaltet?* Es wird von der Anwendung selbst an der Stelle geschrieben, an der die Daten geändert werden. Ein neuer Bildschirm oder ein API-Aufruf ist damit abgedeckt, ohne dass jemand daran denken muss.

Es hat **eine eigene Seite (Admin > Änderungsprotokoll)** und wird auf keiner Aufgaben- oder Projektseite angezeigt. Öffnen können sie nur der **RC Admin** und die Rolle **Auditor**.

1. Was aufgezeichnet wird

Wird aufgezeichnet	Wird nie aufgezeichnet
--------------------	------------------------

Anmeldung: Anmeldung (Passwort, LDAP, Passkey, SSO), Abmeldung, fehlgeschlagene Versuche

Personen und Zugriff: Benutzer angelegt, geändert, aktiviert, deaktiviert, gelöscht; Passwortwechsel; Sicherheitsrollen; Teamzugehörigkeit (hinzugefügt, verschoben, Auslastung, Lead-Kennzeichen)

Einstellungen: jede Änderung der globalen Einstellungen, der Teams, API-Tokens und Webhooks

Aufgaben: angelegt, geändert, erledigt, wieder geöffnet, gelöscht, wiederhergestellt - mit altem und neuem Wert jedes Feldes

Projekte und Meilensteine: Statusänderungen, Einstellungen, Termine, Kosten

Geld und Zeit: Rechnungen (Status, Beträge), geänderte oder gelöschte Zeitbuchungen, Urlaubsanträge und -entscheidungen, Arbeitszeitkorrekturen

Stammdaten: Kunden, Firmen, Definitionen eigener Felder, Projektvorlagen, Basispläne

Daten, die das System verlassen: Rechnungs-Downloads, die Einmalpasswörter eines Sammelimports, Exporte des Änderungsprotokolls selbst

Importe: ein zusammenfassender Eintrag je Importlauf, gestartet, beendet, zurückgenommen

Passwörter, Tokens und andere Geheimnisse (erscheinen als *******)

Der Text von Kommentaren und Beschreibungen - nur dass sich eine Beschreibung geändert hat, und ihre Länge

Name, E-Mail und Telefonnummer einer Person innerhalb einer Änderung - nur deren Länge

Was jemand nur *angesehen* hat

Die eigenen Terminverschiebungen des Planers, Wichtigkeit und Ähnliches

Das **Anlegen** einer Zeitbuchung (jeder Kommentar bucht Minuten); das Ändern oder Löschen wird aufgezeichnet

Jeder Eintrag sagt, **wer** (eine Person, ein API-Token, der MCP-Endpunkt, der Planer oder das System), **wodurch** (Web, REST-API, MCP, Mail, Import, Planer), **von welcher Adresse, was** geschah, **an welchem Objekt** (mit seinem Namen zu dem Zeitpunkt), zu welchem **Team und Projekt** es gehört, und was sich geändert hat als *alt* → *neu*.

Eine **private Aufgabe** hinterlässt nur die Tatsache, dass etwas geschah, nicht ihren Titel oder Inhalt, und der Eintrag ist nur für die Beteiligten sichtbar - auch nicht für einen RC Admin, der nicht dazugehört.

2. Die Seite Änderungsprotokoll

- **Filtern** nach Zeitraum, Wer, Was geschah (eine Aktion wie `task.done` oder ein Anfang wie `user.`), einem Suchtext und der Zahl der Einträge je Seite (25 bis 200). **Zurücksetzen** löscht die Filter.
- **Blättern** mit **Ältere Einträge** und **Neuere Einträge**. Die Seite bleibt schnell, wie groß das Protokoll auch wird.
- **CSV exportieren** oder **JSON exportieren** exportiert, was die aktuellen Filter zeigen – alles, nicht nur die Seite auf dem Bildschirm. Der Export ist selbst ein Eintrag im Protokoll. In der CSV werden Zellen, die wie eine Tabellenformel beginnen, unschädlich gemacht.
- Die Spalte **Einzelheiten** zeigt, was sich geändert hat, als *Feld: alt → neu*.

3. Wer es sehen darf

Nur der **RC Admin** und der **Auditor**. Teamleitungen und Projektleitungen sehen das Protokoll nicht: Es ist kein Management-Bericht. Um jemandem die Rolle Auditor zu geben, öffne den Benutzer in **Admin > Organisation** und füge **Auditor** zu den Zusatzrollen hinzu. Der Auditor kann lesen, filtern, exportieren und die Integrität prüfen; ändern kann er nichts, auch nicht die Aufbewahrung.

4. Integrität: Siegelkette und Ankerkopien

Jeder Eintrag wird in derselben Datenbanktransaktion geschrieben wie die Änderung selbst: Wird die Änderung zurückgerollt, verschwindet auch ihr Eintrag. Etwa eine halbe Minute später **versiegelt** ein Hintergrundjob neue Einträge: Er gibt jedem eine Position und berechnet einen **Schlüssel-Hash** (HMAC-SHA256) über den Eintrag und den Hash des vorigen. Wird später ein Eintrag geändert, entfernt oder eingefügt, passt die Kette ab dieser Stelle nicht mehr. Der Seitenkopf zeigt, wie viele Einträge versiegelt sind und wie viele noch warten.

- **Integrität prüfen** berechnet die ganze Kette neu und nennt den ersten Eintrag, der nicht passt, und warum: geändert, fehlt, falsche Verbindung, fehlt am Ende, oder mit einem anderen Schlüssel versiegelt. Weil es das ganze Protokoll liest, nutze es, wenn es darauf ankommt, nicht jede Minute.

- Die Datenbank **verweigert zusätzlich UPDATE und DELETE** auf versiegelten Einträgen (wo die Datenbank das Einrichten dieses Schutzes erlaubt; die Seite sagt es, wenn nicht).
- **Ankerkopien:** einmal am Tag schreibt die Anwendung eine kleine signierte Aussage - „an Position N hatte die Kette den Hash H“ - in den **Dateispeicher (S3)** oder, wenn kein S3 eingerichtet ist, in einen Ordner auf dem Server. *Integrität prüfen* vergleicht diese Kopien auch mit der Kette. Das fängt jemanden, der Datenbank und Schlüssel hat und die ganze Kette neu schreibt: An die Kopien im Bucket kommt er nicht heran. Nimm einen Bucket, den die Datenbankadministration nicht ändern kann, am besten mit Objektsperre.

Der Audit-Schlüssel. Die Kette wird mit einem Schlüssel geprüft, der nicht in der Datenbank liegt. Die Anwendung nimmt ihn in dieser Reihenfolge: die Umgebungsvariable `RC_AUDIT_KEY`, den Konfigurationswert `audit_key` oder - beim ersten Gebrauch automatisch angelegt - ein Objekt im **S3-Bucket** des Dateispeichers (`<Präfix>/audit/audit.key`). Nur wenn kein S3 eingerichtet ist, bleibt die Datei `userdata/audit.key`; die Seite zeigt dann eine Warnung, denn auf einer Platte, die nicht erhalten bleibt (Container, mehrere Replikate), finge jeder Redeploy einen neuen Schlüssel an, und die davor versiegelten Einträge ließen sich nicht mehr prüfen. Mit S3 als Dateispeicher übersteht der Schlüssel Redeployments und gilt für alle Replikate und den Planer. Für strengere Trennung setze stattdessen `RC_AUDIT_KEY` in den Geheimnissen deines Deployments. **Ohne den Schlüssel lässt sich die Kette nicht prüfen.** Deshalb fordert die Seite den RC Admin auf, den **Schlüssel herunterzuladen** (der Auditor darf es auch; jeder Download ist ein Eintrag) und zu bestätigen, dass eine Kopie aufbewahrt wird. Ein älterer Schlüssel lässt sich **hinzufügen** (im Bucket unter `audit/keys/` abgelegt), ein verlorener lässt sich **als verloren akzeptieren**: Seine Einträge werden dann weiter auf Reihenfolge und Verbindungen geprüft, nicht auf den Inhalt, und die Prüfung meldet, wie viele (`unverified`). Die Prüfung wählt den Schlüssel nach der `key_id` des Eintrags.

Was das ehrlich leistet: Jede Änderung am Protokoll durch jemanden ohne den Schlüssel ist **erkennbar**; mit den Ankerkopien auch das Neuschreiben durch jemanden, der alles außer dem Bucket hat. Das ist *manipulationssicher nachweisbar*, nicht *manipulationssicher*.

5. Aufbewahrung und Datenschutz

Ein RC Admin sieht die Karte **Aufbewahrung und Datenschutz**.

- **Einträge aufbewahren (Jahre)** - voreingestellt sind 10, der übliche Zeitraum für Geschäftsunterlagen; **0 bewahrt unbegrenzt auf**. Einmal am Tag werden ältere Einträge vom ältesten Ende der Kette her entfernt, in ganzen Bereichen; der Rest lässt sich weiter prüfen, und das Entfernen ist ein eigener Eintrag. Auch das Ändern der Einstellung ist ein Eintrag.
- **Person pseudonymisieren** - für ein Löschersuchen. Gib den Benutzernamen ein; nach einer Bestätigung zeigt jeder Eintrag von oder über diese Person ein Pseudonym (`pseudonym-ab12cd34`) statt des Namens. Dieselbe Person bekommt (solange der Schlüssel derselbe bleibt) dasselbe Pseudonym, ihre Einträge gehören also weiter zusammen, der Name lässt sich aber nicht zurücklesen. Es lässt sich **nicht rückgängig machen**, es funktioniert auch für gelöschte Konten, und die Kette bleibt gültig. Name, E-Mail und Telefonnummer einer Person stehen ohnehin nicht in einer Änderung in den Einträgen.

Für den Betriebsrat: Das Protokoll hält **Änderungen** fest, die Personen an Daten und Einstellungen vornehmen, nicht, was sie angesehen haben, und nicht ihre Arbeitszeit. Es enthält aber, wer was getan hat. Sage deinem Betriebsrat und deiner Datenschutzbeauftragten, dass es das gibt, wer es öffnen kann und wie lange es aufbewahrt wird; die Aufbewahrungseinstellung und das Pseudonymisieren sind für dieses Gespräch da.

6. REST-API und MCP

Nur lesend, dieselbe Zugriffsregel wie die Seite (RC Admin oder Auditor; alle anderen bekommen 403). Keine API kann in das Protokoll schreiben.

Aufruf	Was er tut
<code>GET /api/v1/audit-log</code> · MCP <code>list_audit_log</code>	Einträge, neueste zuerst. Filter: <code>from</code> , <code>to</code> , <code>actor</code> , <code>action</code> , <code>object_type</code> , <code>object_id</code> , <code>team_id</code> , <code>project_id</code> , <code>q</code> ; <code>limit</code> (höchstens 200). Die Antwort enthält <code>next</code> (als <code>before</code> für ältere Einträge) und <code>prev</code> (als <code>after</code> für neuere).
<code>GET /api/v1/audit-log/verify</code> · MCP <code>verify_audit_log</code>	Zahlen, Aufbewahrung, Ankerstatus und das Ergebnis der Neuberechnung der Kette und des Vergleichs der Ankerkopien. Liest das ganze Protokoll.

Ein Token mit nur Leserechten genügt. Die vollständige Beschreibung steht in `api/openapi.yaml`.

7. Die Kette selbst prüfen

Eine Prüferin muss dem Knopf „Integrität prüfen“ nicht vertrauen. Für einen Eintrag an Position n mit dem Hash des vorigen Eintrags p (64 Nullen beim ersten):

`hash = HMAC-SHA256(Schlüssel, p + "|" + canonical)`, wobei `canonical` das JSON-Array dieser Felder in dieser Reihenfolge ist: `id, seq, at, actor_id, actor_type, via, ip, request_id, action, object_type, object_id, team_id, project_id, changes, meta, restricted_to` (jeweils als Text, `null` bleibt `null`; JSON ohne Maskieren von Schrägstrichen und Nicht-ASCII-Zeichen). Der Name des Handelnden und der Name des Objekts sind **nicht** Teil des Hashs - dadurch lässt sich eine Person pseudonymisieren, ohne die Kette zu brechen. Jeder versiegelte Eintrag im JSON-Export und in der API-Antwort hat einen Block `chain` mit den Rohwerten, über die der Hash gebildet wurde (`at` mit Mikrosekunden, `changes` und `meta` als gespeicherter Text, `seq`, `prev_hash`, `row_hash`); die CSV enthält `seq` und `row_hash`. Die Funktion `rc_audit_row_hash()` in `app_libs/audit.func.php` ist die Referenz.

Regeln und Grenzen

- Einträge der letzten halben Minute sind noch nicht versiegelt; die Seite sagt, wie viele warten.
- Exporte, die ein Browser selbst erzeugt (die Export-Knöpfe der Tabellen auf Berichtsseiten), erreichen den Server nie und werden nicht aufgezeichnet; Downloads, die der Server erzeugt (Rechnungen, Passwörter des Sammelimports, das Änderungsprotokoll), schon.
- Änderungen direkt in der Datenbank, an der Anwendung vorbei, werden nicht aufgezeichnet - aber ein fehlender oder veränderter Eintrag in der Kette und die Ankerkopien zeigen, dass das Protokoll selbst angefasst wurde.
- Das Protokoll beginnt mit der Installation der Funktion; ältere Geschichte bleibt in den älteren Protokollen je Bereich (Projektprotokoll, Story-Protokoll, Rechnungsereignisse).
- Einträge verfallen nach der Aufbewahrungszeit. Ankerkopien des entfernten Teils werden bei der Prüfung übergangen.
- Die Entwicklungs- und Testdatenbanken füllen sich mit Testeinträgen; das Protokoll lässt nur Anhängen zu, sie bleiben also.