



TraxrGRC

Administrator's Guide

October 2026 · build 244823a

Installing, configuring, securing and operating a TraxrGRC instance.

About this guide

This guide is for the people who install and run TraxrGRC: the administrator who configures users, roles, sign-in and settings in the application, and the operator who runs the server, its backups and its upgrades. In a small team these are often the same person.

Who this guide is for

- **Application administrators** who hold the Administrator role and manage users, groups, roles, single sign-on, settings, notifications, data loading and API access.
- **Operators** who install the platform on a server, keep it backed up, apply upgrades and watch its health. Operator chapters assume you are comfortable with a Linux shell, Docker Compose and a text editor.
- **Security and audit staff** who need to know what the platform enforces, what it records and how to show that the record is intact.

People who use the modules day to day should read the User Manual. New users should start with the Quick Start Guide.

How this guide is organized

Chapter	What it covers
Architecture and Requirements	The components of an instance, how they connect, and what the server needs
Installation	Installing from a release, HTTPS options, the first administrator and the go-live checklist
Configuration Reference	Every environment variable and where each setting lives
Users, Groups and Roles	Accounts, groups, roles and permissions, own and all scope, separation of duties
Authentication and Single Sign-On	Passwords, throttling, LDAP, SAML, SCIM provisioning and sessions
Platform Settings	Every Settings page: branding, labels, custom fields and the module settings
Notifications and Email	SMTP, templates, rules, the scheduler, scheduled reports and the delivery log
Loading and Importing Data	CSV imports, framework packages, the bundle loader, first-time migration and demo data
API Access and Integrations	API tokens, the REST API, AI assistants (MCP) and service accounts
Audit Log and Evidence Integrity	What is logged, the Audit Log page and the tamper-evident hash chain
Backup and Restore	Server backups, in-app snapshots and a restore drill
Upgrades	Applying a new release, what the update checks, and rolling back
Monitoring and Troubleshooting	Health checks, logs, chain verification and fixes for common failures
Security Hardening	What the platform enforces and the operator's checklist

A first installation follows the chapters in order through Platform Settings. After go-live, the last four chapters are the routine.

Conventions

Convention	Meaning
Bold	A label you see on screen: a menu entry, page title, tab, button, field or status, written exactly as it appears.
Settings → Integrations	A navigation path: open Settings in the sidebar, then Integrations .
<code>monospace</code>	Text you type or read literally: addresses, file names, commands, environment variables, permission names and API values.
<code>module:right</code>	A permission, for example <code>settings:update</code> (Update on the Settings module).
<i>own and all</i>	The two levels at which a role can hold a right; see Users, Groups and Roles.
Command blocks	Commands to run on the server, from the instance directory (the one holding <code>docker-compose.yml</code>) unless the text says otherwise.

Settings pages. Every Settings page has a card on the Settings page, and each card is shown only to people the page itself admits. The sidebar's **Settings** entry appears for anyone who can open at least one of them. Each chapter also gives the page's address.

Screenshots show a demonstration installation with fictional data for an invented company, Quillmont Systems. Your product name, logo and data will differ.

Contents

1	Architecture and Requirements	9
1.1	Components	9
1.2	How the components connect	11
1.3	Host requirements	14
1.4	Supported browsers	14
1.5	Network ports	15
2	Installation	16
2.1	Before you begin	16
2.2	Unpack the release	16
2.3	Create the .env file	17
2.4	Choose how HTTPS is provided	18
2.5	First start	20
2.6	Bootstrapping the first administrator	21
2.7	First sign-in	22
2.8	Post-installation checklist	24
2.9	Running several instances on one server	24
3	Configuration Reference	26
3.1	How values reach the containers	26
3.2	Instance settings (.env)	27
3.3	Set by Compose and the image	32
3.4	Operator script variables	33
3.5	Development and demonstration only	34
3.6	Settings that live in the application	35
4	Users, Groups and Roles	36
4.1	Where identity is administered	36
4.2	User accounts	38
4.3	Groups	43
4.4	Roles	46
4.5	Separation of duties	54
4.6	Staff and portal-only users	56
4.7	Effect of changes on open sessions	57
4.8	Permission reference	58
5	Authentication and Single Sign-On	64
5.1	Sign-in methods at a glance	64

5.2	Local passwords	64
5.3	Sign-in throttling and logout	66
5.4	Two-step verification	67
5.5	LDAP	69
5.6	SAML single sign-on	72
5.7	Account linking	78
5.8	SSO only and break-glass administrators	78
5.9	SCIM provisioning	79
5.10	Sessions and sign-out	83
6	Platform Settings	85
6.1	How settings are stored and applied	85
6.2	Map of the settings pages	85
6.3	Branding	90
6.4	Labels	93
6.5	Custom fields	94
6.6	Risk Settings	98
6.7	Control testing settings	101
6.8	Incident classifications	102
6.9	Policy settings	102
6.10	Privacy settings	103
6.11	Sessions page: token lifetime, storage and separation of duties	105
6.12	Settings checklist for a new instance	106
7	Notifications and Email	108
7.1	How email is sent	108
7.2	Configuring SMTP	108
7.3	Testing email	110
7.4	The Notifications page	111
7.5	Templates	111
7.6	Scheduler	118
7.7	Notification rules	121
7.8	Scheduled reports delivery	123
7.9	The delivery log	124
7.10	Deliverability	125
8	Loading and Importing Data	127
8.1	Choosing a method	127
8.2	What a new installation already contains	128
8.3	CSV imports	128
8.4	Compliance framework packages	132

8.5	The bundle loader	134
8.6	First-time migration	137
8.7	Demo data	138
9	API Access and Integrations	140
9.1	Integration points	140
9.2	API tokens	141
9.3	The REST API for administrators	143
9.4	AI assistants (MCP)	144
9.5	Service accounts and token practice	147
10	Audit Log and Evidence Integrity	149
10.1	What is logged	149
10.2	The Audit Log page	151
10.3	The hash chain	154
10.4	Retention	156
10.5	Security alerts	156
10.6	Evidence file fingerprints	158
10.7	Files moved onto evidence	158
10.8	Using the audit log as evidence	158
11	Backup and Restore	160
11.1	Server backups	160
11.2	Restoring a server backup	162
11.3	In-app snapshots	164
11.4	Restore drill	167
12	Upgrades	170
12.1	Before you upgrade	170
12.2	Running an update	170
12.3	What the update does	171
12.4	When an update fails	173
12.5	Rolling back	173
12.6	Cleaning up after upgrades	174
12.7	Upgrading an installation from before October 2026	174
13	Monitoring and Troubleshooting	176
13.1	Health checks	176
13.2	Logs	177
13.3	Row counts	178
13.4	Common problems	179

14	Security Hardening	183
14.1	What the product enforces	184
14.2	Operator checklist	185
14.3	Known limits	188
	Index	

Architecture and Requirements

TraxrGRC is a self-hosted web application that runs as a small set of Docker containers on one server. Each organization gets its own instance: its own application, background worker, database and file store. Nothing is shared between instances, and there is no central service that an instance calls home to.

This chapter describes the parts of an instance, how they connect, and what the server and the network must provide. Read it before you install, and keep it at hand when you plan firewall rules, monitoring and backups.

1.1 Components

An instance is defined by `docker-compose.yml` in the release directory. One application image, built from the release's `Dockerfile`, runs in three roles (`migrate`, `web` and `worker`); the other services use public images.

Service	Image	What it does	Runs
<code>db</code>	<code>postgres:18-alpine</code>	The PostgreSQL 18 database holding every record, the audit log and the background job queue.	Always. Not published outside the Docker network.
<code>migrate</code>	Application image, role <code>migrate</code>	Applies database migrations, creates or updates the application's database role and its grants, adds missing reference data, encrypts any stored secret still in plain text and, on an empty database only, creates the first administrator. Then exits.	Once per <code>docker compose up</code> , before <code>web</code> and <code>worker</code> .
<code>web</code>	Application image, role <code>web</code>	The Next.js application server on port 3000: every page, the REST API, SCIM, the MCP endpoint and <code>/api/health</code> .	Always.
<code>worker</code>	Application image, role <code>worker</code>	Background jobs: email delivery, daily reminder sweeps, notification rules, scheduled reports and the weekly in-app backup snapshot.	Always.
<code>caddy</code>	<code>caddy:2.11-alpine</code>	Optional HTTPS reverse proxy with automatic certificates.	Only with <code>--profile tls</code> .
<code>mailpit</code>	<code>axllent/mailpit:v1.21</code>	A local mail catcher for trying out notifications. Never use it in production.	Only with <code>--profile dev</code> .

The instance keeps its state in four named Docker volumes. Compose prefixes each with the project name (`grc` unless you set `COMPOSE_PROJECT_NAME`), so the uploads volume of the default project is `grc_uploads`.

Volume	Mounted in	Contents
<code>pgdata18</code>	<code>db</code> at <code>/var/lib/postgresql</code>	The database.
<code>uploads</code>	<code>web</code> and <code>worker</code> at <code>/app/uploads</code>	Uploaded attachments (including policy documents and evidence), and the weekly in-app snapshots in its <code>backups</code> subdirectory.
<code>caddy-data</code>	<code>caddy</code>	Certificates and the ACME account.
<code>caddy-config</code>	<code>caddy</code>	Caddy's saved configuration.

The database and the uploads volume are everything you must back up. The two Caddy volumes can be recreated: Caddy requests new certificates if they are lost.

1.2 How the components connect

The containers share a private Docker network. Only two things listen on the host: the `web` port (on `127.0.0.1:3000` by default) and, with the `tls` profile, Caddy on ports 80 and 443.

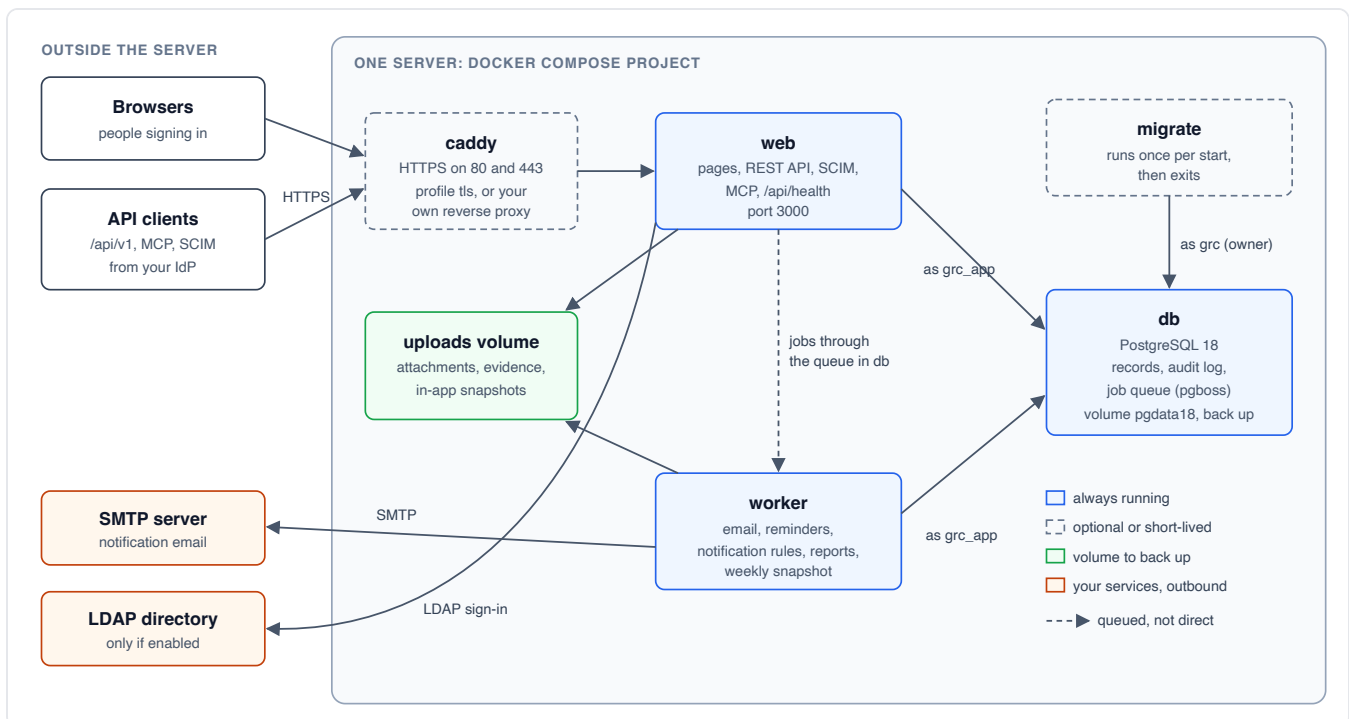


Figure 1.1. How an instance's containers, volumes and outside services connect.

1. A browser or API client connects to Caddy (or your own reverse proxy) over HTTPS. The proxy forwards the request to `web:3000`.
2. `web` reads and writes the database as the application role. It never writes files anywhere except the `uploads` volume and two in-memory directories (`/tmp` and its Next.js cache).
3. When something has to happen later or outside the request (an email, a notification rule), `web` puts a job on the queue. The queue is pg-boss, stored in the database's own `pgboss` schema, so there is no separate message broker.
4. `worker` takes jobs from the queue, runs the scheduled tasks, and sends email through the SMTP server configured under **Settings → Integrations**.

Start-up order is enforced by Compose: `migrate` waits for `db` to report healthy, and `web` and `worker` start only after `migrate` has finished successfully. If `migrate` fails, `web` and `worker` do not start, so the application never runs against a database it has not prepared.

1.2.1 Database roles

The database has two login roles, and the split is part of how the audit log is protected.

Role	Name (default)	Password	Used by
Owner	<code>POSTGRES_USER</code> (<code>grc</code>)	<code>POSTGRES_PASSWORD</code>	<code>migrate</code> , and <code>deploy/backup.sh</code> and <code>deploy/restore.sh</code> , which run inside the <code>db</code> container. Owns every table.
Application role	<code>APP_DB_USER</code> (<code>grc_app</code>)	<code>APP_DB_PASSWORD</code>	<code>web</code> and <code>worker</code> .

The application role can read and write the application's tables, but on the audit log it can only read and add entries; it owns nothing in the application's schema and so cannot alter, disable or drop the audit log's triggers. It owns the `pgboss` schema, which the queue needs. Neither `web` nor `worker` is given the owner's password.

`migrate` creates the application role on the first start and, on every start, sets its password from `APP_DB_PASSWORD` and reapplies its grants. A changed password or a new table needs nothing more than `docker compose up -d`.

1.2.2 Background jobs

The worker registers these queues and schedules. All schedules are in UTC.

Queue	When	What it does
<code>send-email</code>	As jobs arrive	Sends one queued notification email.
<code>send-rule-email</code>	As jobs arrive	Sends one email produced by a notification rule, with retries.
<code>daily-sweeps</code>	The digest schedule set under Settings → Notifications (default <code>0 7 * * *</code> , 07:00 UTC daily)	Marks lapsed risk, compliance and policy exceptions as expired (audited), then review and deadline reminders and digests, deadline-based notification rules, online assessment repeats and reminders, the POA&M sync from requirement status, and policy acknowledgement reminders.
<code>notification-rules</code>	Every minute	Reads new audit log entries for notification rules and recovers stuck rule deliveries.
<code>report-schedules</code>	Every minute	Runs scheduled reports that are due.
<code>weekly-backup</code>	<code>0 3 * * 0</code> (Sundays at 03:00 UTC)	Writes a signed in-app snapshot to the uploads volume and keeps the newest four.

Saving the digest schedule under Settings → Notifications applies it at once; no restart is needed. If the save cannot reach the job queue (for example because the worker has never run), the page says so and the worker picks up the saved schedule when it next starts.

Jobs wait in the database while the worker is stopped and run when it comes back, so stopping the worker delays email; it does not lose it.

1.2.3 Container hardening

`web`, `worker` and `migrate` run as an unprivileged user (uid 100, gid 101) with a read-only root filesystem, no Linux capabilities and `no-new-privileges`. The application code in the image is owned by root, so even the application cannot change it. The only writable places are the `uploads` volume, `/tmp` and, for `web`, `/app/.next/cache` (both in memory and emptied on restart). The bundled Caddy runs with the same settings, keeping only the capability to listen on ports below 1024. The image contains production dependencies only.

1.3 Host requirements

Item	Requirement
Operating system	A Linux server (x86-64 or ARM64) with Docker Engine and the Docker Compose v2 plugin (the <code>docker compose</code> command).
CPU and memory	2 vCPU and 4 GB of RAM are enough for most organizations. Building the image during installation and upgrades is the most demanding step.
Disk	20 GB to start. Plan for the database, uploaded files, the Docker images (each upgrade builds a new image and keeps the previous one for rollback), and local backup copies if you keep them on the same server.
DNS	A host name of its own for the application, for example <code>grc.example.com</code> , pointing at the server. Do not share the name with other services (see Security Hardening, HSTS).
Host tools	<code>sh</code> , <code>tar</code> , <code>find</code> and <code>openssl</code> . For encrypted backups, <code>age</code> .
Time	The server clock synchronized (NTP). Session expiry, SAML assertion checks, sign-in throttling and every schedule depend on it.

1.4 Supported browsers

TraxrGRC is a Next.js 16 application and supports the browsers Next.js supports: current versions of Chrome, Edge and Firefox (version 111 or later) and Safari 16.4 or later, on desktop. The product's automated end-to-end tests run in Chromium and Firefox. JavaScript and cookies must be enabled.

1.5 Network ports

Port	Direction	Used for	Notes
443/tcp	Inbound	HTTPS to the application	Caddy with <code>--profile tls</code> , or your own proxy.
80/tcp	Inbound	HTTP to HTTPS redirect and the ACME HTTP challenge	Bundled Caddy only.
3000/tcp	Host loopback	<code>web</code> , for the reverse proxy	Bound to <code>127.0.0.1</code> by default (<code>WEB_BIND</code> , <code>WEB_PORT</code>). Never publish it to the network without a proxy (see Security Hardening).
5432/tcp	Docker network only	PostgreSQL	Never published on the host.
8025/tcp	Host loopback	Mailpit web interface	<code>--profile dev</code> only.
25, 465 or 587/tcp	Outbound	SMTP to your mail relay	Whatever port the relay uses, set under Settings → Integrations.
636 or 389/tcp	Outbound	LDAP directory	Only if LDAP sign-in is enabled. <code>ldaps://</code> (636) or <code>ldap://</code> with StartTLS (389).
443/tcp	Outbound	Let's Encrypt (ACME)	Bundled Caddy only, to obtain and renew certificates.
443/tcp	Outbound	<code>login.microsoftonline.com</code>	Only if AI assistant access with Microsoft Entra ID sign-in is configured.
443/tcp	Outbound, during builds	Docker Hub and the npm registry	The image is built on the server at installation and on every upgrade.

SAML single sign-on needs no outbound connection from the server: the identity provider's response travels through the user's browser.

Installation

This chapter installs one instance with Docker Compose. Installation takes about fifteen minutes, most of it the first image build.

2.1 Before you begin

You need:

- A server that meets the host requirements in Architecture and Requirements, and an account on it that can run `docker`.
- The DNS name for the instance, already pointing at the server.
- The TraxrGRC release archive.
- If you use the bundled HTTPS proxy: ports 80 and 443 reachable from the internet (or from Let's Encrypt) and an email address for certificate notices.
- The address, port and credentials of an SMTP relay. Security alerts reach administrators only by email, so configure SMTP before go-live.

A release is the product's source tree: `docker-compose.yml`, the `Dockerfile`, the `deploy/` scripts and the application code. Compose builds the application image from it on your server.

2.2 Unpack the release

1. Create the installation directory and unpack the release into it. This guide uses `/srv/grc` throughout.

```
sudo mkdir -p /srv/grc
sudo chown "$USER" /srv/grc
tar -xzf grc-release-<version>.tar.gz -C /srv/grc
cd /srv/grc
```

2. Check that `docker-compose.yml`, `Dockerfile` and the `deploy/` directory are at the top of `/srv/grc`. Every command in this guide runs from that directory.

Keep the installation directory for this release only. Upgrades replace its contents (see Upgrades), and a file you add there yourself stops the next upgrade unless you list it in `.deploy-keep`.

2.3 Create the .env file

All instance configuration that is not made in the application lives in `.env` next to `docker-compose.yml`.

1. Copy the example and restrict it to your account:

```
cp deploy/env.example .env
chmod 600 .env
```

2. Generate the three required secrets and paste each into `.env`:

```
openssl rand -base64 32 # AUTH_SECRET
openssl rand -hex 32    # POSTGRES_PASSWORD
openssl rand -hex 32    # APP_DB_PASSWORD
```

Use hexadecimal for the two database passwords: they go into a database connection URL, where characters such as `/`, `@` and `+` would break it.

3. Fill in the remaining values:

Setting	Value
<code>APP_URL</code>	The public address users reach, with <code>https://</code> and no trailing slash, for example <code>https://grc.example.com</code> .
<code>APP_NAME</code>	The product name shown until an administrator sets one under Settings → Branding.
<code>INITIAL_ADMIN_EMAIL</code>	The email address of the first administrator.
<code>INITIAL_ADMIN_PASSWORD</code>	Its password, 12 characters or more.
<code>DOMAIN</code>	With the bundled Caddy: the host name only, for example <code>grc.example.com</code> .
<code>ACME_EMAIL</code>	With the bundled Caddy: the address Let's Encrypt sends certificate notices to.

4. Decide now whether to set `BACKUP_SIGNING_KEY` and `SETTINGS_ENCRYPTION_KEY` (32 characters or more each, for example from `openssl rand -base64 32`). Both are optional. Setting them at installation lets you rotate `AUTH_SECRET` later without losing in-app restore of older snapshots or the stored integration passwords. Adding them later has the same effect as changing `AUTH_SECRET`. See Configuration Reference.

5. Store a copy of `.env` in your organization's password vault.

Warning: Losing `AUTH_SECRET` signs everyone out and makes the stored SMTP, LDAP and SAML secrets unreadable; losing `POSTGRES_PASSWORD` locks you out of the database owner's account.

The full list of settings, with defaults, is in Configuration Reference. Compose reads `.env` only to fill in `docker-compose.yml`: a variable reaches the containers only if `docker-compose.yml` passes it on, so adding other variables to `.env` has no effect on the application.

2.4 Choose how HTTPS is provided

Pick one of the three options below before the first start.

2.4.1 Option 1: the bundled Caddy

The `tls` profile starts Caddy in front of `web`. Caddy obtains and renews a certificate from Let's Encrypt for `DOMAIN`, redirects HTTP to HTTPS, sends the HSTS header, compresses responses, and refuses request bodies larger than the application accepts (32 MiB, and 2049 MiB on `/api/backups/restore`) before they reach it. Caddy also replaces any `X-Forwarded-For` header a client sends with the address it saw, so the default `TRUSTED_PROXY_HOPS=1` is correct.

Requirements: `DOMAIN` and `ACME_EMAIL` set in `.env`, ports 80 and 443 free on the host and reachable for the ACME challenge. Caddy refuses to start without `DOMAIN`.

Start every command that brings the stack up with `--profile tls`.

2.4.2 Option 2: the bundled Caddy with your own certificate

If the server cannot be reached by Let's Encrypt (an internal network), keep the bundled Caddy and give it a certificate you supply. Do not edit `deploy/Caddyfile`: an upgrade replaces it. Instead:

1. Copy the shipped file to a top-level `Caddyfile.*` name, which upgrades never touch:

```
cp deploy/Caddyfile Caddyfile.local
```

2. In `Caddyfile.local`, add a `tls` line inside the site block that names your certificate and key as they will appear inside the container:

```
{${DOMAIN}} {
    tls /certs/fullchain.pem /certs/privkey.pem
    ...
}
```

For a server reached only on an internal network, `tls internal` makes Caddy issue its own certificate from a local authority instead; browsers then need Caddy's root certificate installed to trust it.

3. Create `docker-compose.override.yml` (also never touched by upgrades) to mount the file and the certificates:

```
services:
  caddy:
    volumes:
      - ./Caddyfile.local:/etc/caddy/Caddyfile:ro
      - /etc/ssl/grc:/certs:ro
```

4. Keep the request body limits and the `reverse_proxy web:3000` line from the shipped file. Compare your copy with `deploy/Caddyfile` after each upgrade and carry over any change.

2.4.3 Option 3: your own reverse proxy

Leave out `--profile tls` and point your proxy at `http://127.0.0.1:3000`. The proxy must:

- Pass the `Host` header the browser sent (or set `X-Forwarded-Host`). The application compares it with the request's `Origin` to refuse cross-site requests.
- Overwrite `X-Forwarded-For` with the client's address, or append to it. Set `TRUSTED_PROXY_HOPS` in `.env` to the number of proxies that append, for example `2` for a cloud load balancer in front of nginx. Sign-in throttling and the address recorded on every audit entry depend on this.
- Cap request bodies at 32 MiB, and at 2049 MiB for `/api/backups/restore`.
- Allow at least five minutes for a response on `/api/backups/restore` and `/api/backups/export`; both work on the whole database.
- Send `Strict-Transport-Security` or leave the application's own header in place.

An nginx example:

```

server {
    listen 443 ssl;
    server_name grc.example.com;
    ssl_certificate      /etc/ssl/grc/fullchain.pem;
    ssl_certificate_key  /etc/ssl/grc/privkey.pem;

    client_max_body_size 32m;

    location = /api/backups/restore {
        client_max_body_size 2049m;
        proxy_read_timeout 300s;
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location / {
        proxy_read_timeout 300s;
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

If the proxy runs on another machine, set `WEB_BIND` in `.env` to the server's private address (never `0.0.0.0` on a host with a public interface) and firewall port 3000 so that only the proxy can reach it.

2.5 First start

1. Build the image and start the stack. With the bundled Caddy:

```
docker compose --profile tls up -d --build
```

Behind your own proxy, leave out `--profile tls`:

```
docker compose up -d --build
```

2. Follow the `migrate` service until it exits:

```
docker compose logs -f migrate
```

On a new installation the log ends with lines like these:

```
app-role: grc_app created; grants applied; pgboss owned by grc_app.  
[bootstrap] reference data added: <n> roles, <n> classifications, <n> templates, <n> data  
[bootstrap] shared dashboard: default layout created  
[bootstrap] created administrator admin@example.com; remove INITIAL_ADMIN_PASSWORD now
```

<n> stands for the counts, which depend on the release. Before them, Prisma lists the database migrations it applied.

3. Check that every service is up and that `web` reports healthy:

```
docker compose ps
```

`migrate` shows as exited with code 0; `db` and `web` show `(healthy)`; `worker` (and `caddy`) show as running.

4. Check the health endpoint:

```
curl -s http://127.0.0.1:3000/api/health  
# {"status":"ok"}  
curl -s https://grc.example.com/api/health  
# {"status":"ok"}
```

If `migrate` stops with an error, `web` and `worker` stay down; see Monitoring and Troubleshooting.

2.6 Bootstrapping the first administrator

The first administrator is created by the bootstrap step that `migrate` runs on every start (`prisma/bootstrap.ts`, available in development as `npm run db:bootstrap`). The bootstrap:

1. Adds any missing reference data (default roles, classifications, email templates and the default data mapping questionnaire). It never changes or deletes existing data, so it is safe on every start, with one exception: the shared dashboard. When a release ships a new default layout, the bootstrap replaces the shared dashboard with it if nobody has changed it since the last default was applied (personal dashboards are never touched; their owners get a "New widgets available" note instead), writes an audit entry (module dashboard, entity dashboardLayout), and logs `[bootstrap] shared dashboard: default layout updated`. A dashboard someone has edited (a widget added, removed, moved or retitled) is left as it is, logged as `edited by an`

administrator, default layout not applied. The layout last applied is recorded in the dashboard.appliedDefaultLayout setting, which travels with backups.

2. Encrypts any secret setting still stored in plain text.
3. If the database has no users at all, creates an administrator from INITIAL_ADMIN_EMAIL and INITIAL_ADMIN_PASSWORD. The address is stored in lower case, the name is set to Administrator, and the account is marked **break-glass**, so it keeps password sign-in if you later turn on **SSO only**.

When users already exist, the bootstrap logs [bootstrap] users exist; initial administrator skipped and ignores both variables. When no users exist and either variable is missing, or the password is shorter than 12 characters, it fails with:

```
[bootstrap] No users exist. Set INITIAL_ADMIN_EMAIL and INITIAL_ADMIN_PASSWORD (at least 12 c
```

migrate then exits with an error and the application does not start. Set both values and run docker compose up -d again.

The application never reads the password from anywhere except INITIAL_ADMIN_PASSWORD. Some operators record the initial administrator's password in a file named .initial-admin (mode 600) in the installation directory; upgrades leave a file of that name alone, but the safer place is your password vault.

2.7 First sign-in

1. Open APP_URL in a browser. You are taken to the sign-in page.
2. Enter the initial administrator's **Email** and **Password** and click **Sign in**.

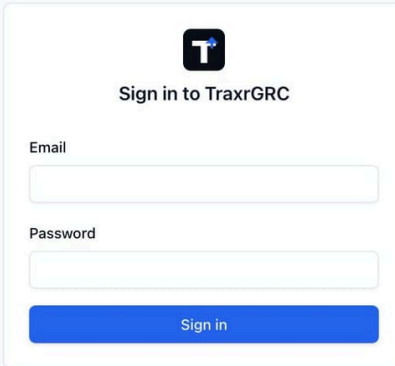
The image shows a sign-in page for TraxrGRC. At the top center is a logo consisting of a black square with a white 'T' and a blue plus sign. Below the logo is the text 'Sign in to TraxrGRC'. Underneath this are two input fields: one labeled 'Email' and one labeled 'Password'. At the bottom of the form is a blue button with the text 'Sign in'.

Figure 2.1. The sign-in page with the Email and Password fields and the Sign in button.

3. Open the user menu, choose **My profile**, and set your **Time zone**.
4. Change the initial password: in **Settings → Users**, open the Administrator account and set a new password (see Users, Groups and Roles). Store it in the vault.

Then remove the initial password from the server:

1. Delete the value of `INITIAL_ADMIN_PASSWORD` in `.env` (leave the line as `INITIAL_ADMIN_PASSWORD=`).
2. Recreate the exited `migrate` container, which otherwise keeps the old value in its configuration:

```
docker compose up -d --no-deps --force-recreate migrate
```

3. Confirm the value is gone:

```
docker inspect "$(docker compose ps -aq migrate)" | grep INITIAL_ADMIN_PASSWORD
```

The command should print `"INITIAL_ADMIN_PASSWORD="` with no value.

2.8 Post-installation checklist

Work through this list before you invite users. The chapters named in each row describe the screens.

Step	Where	Chapter
Initial password changed, removed from <code>.env</code> and from the <code>migrate</code> container.	Server, Settings → Users	This chapter
A copy of <code>.env</code> stored in the password vault.	Vault	Installation
Product name, logo and color set.	Settings → Branding	Platform Settings
SMTP configured and Send test email received.	Settings → Integrations	Notifications and Email
Session length and idle timeout reviewed.	Settings → Sessions	Authentication and Single Sign-On
LDAP or SAML configured and tested, if used.	Settings → Integrations	Authentication and Single Sign-On
A second administrator created and marked break-glass, password in the vault.	Settings → Users	Users, Groups and Roles
SSO only turned on, if every user signs in through the identity provider.	Settings → Integrations	Authentication and Single Sign-On
Groups and roles set up for least privilege.	Settings → Groups, Settings → Roles	Users, Groups and Roles
Nightly <code>deploy/backup.sh</code> scheduled, copied off the server, and one restore drill passed.	Server	Backup and Restore
Daily audit chain verification scheduled.	Server	Monitoring and Troubleshooting
External monitoring of <code>/api/health</code> .	Monitoring system	Monitoring and Troubleshooting
Existing registers loaded, if any.	API	Loading and Importing Data

2.9 Running several instances on one server

Each instance needs its own directory, `.env`, project name and volumes. In each instance's `.env`, set:

- `COMPOSE_PROJECT_NAME` to a distinct name, so its volumes and containers do not collide with another instance's. The application image is named after it (`grc-platform-
<COMPOSE_PROJECT_NAME>:latest`), so each instance builds, runs and rolls back its own image.
- `WEB_PORT` to a distinct port.

If you set `APP_IMAGE` yourself, give each instance a different value; two instances sharing one image name would build over and roll back each other's image.

Only one process can listen on ports 80 and 443, so at most one instance can use the bundled Caddy's default configuration. Put one reverse proxy in front of all instances, or give each instance's Caddy its own port in a `Caddyfile.*` and `docker-compose.override.yml`.

Configuration Reference

TraxrGRC takes its server-level configuration from environment variables set in `.env`. Everything else (email, sign-in integrations, sessions, branding, notifications) is configured in the application under Settings and stored in the database.

3.1 How values reach the containers

Compose reads `.env` to fill in `docker-compose.yml`. Each container receives only the variables `docker-compose.yml` lists for it, sometimes under another name (for example, `APP_URL` is also passed as `AUTH_URL`). Setting a variable in `.env` that `docker-compose.yml` does not use changes nothing. To pass anything else, add it to the service's `environment` in `docker-compose.override.yml`.

After changing `.env`, apply it with the same command you start the stack with, for example:

```
docker compose --profile tls up -d
```

Compose recreates the containers whose configuration changed; `migrate` runs again first.

In the tables below, **Required** means Compose or the application refuses to start without the value. **Secret** values must be kept out of tickets, chat and source control.

3.2 Instance settings (.env)

Variable	Purpose	Default	Require
Core			
<code>APP_URL</code>	The public address of the instance, with scheme and no trailing slash. Used for sign-in redirects (passed to Auth.js as <code>AUTH_URL</code>), links in emails, the SAML service provider addresses, SCIM and MCP addresses, and as an accepted origin for cross-site request checks. Use <code>https://</code> ; SAML sign-in requires it.	None	Yes (Comp
<code>APP_NAME</code>	Product name shown until one is set under Settings → Branding.	<code>GRC Platform</code>	No
<code>AUTH_SECRET</code>	Signs session tokens and the SAML sign-in state. Unless the dedicated keys below are set, it is also the source of the backup signing key and the key that encrypts stored integration secrets. <code>web</code> refuses to start in production when it is missing,	None	Yes (Comp and <code>w</code>

Variable	Purpose	Default	Require
	shorter than 32 characters, or contains a placeholder (<code>change-me</code> , <code>changeme</code> or <code>dev-secret</code>).		
<code>BACKUP_SIGNING_KEY</code>	Source of the key that signs in-app backup snapshots. When empty, the key is derived from <code>AUTH_SECRET</code> . At least 32 characters when set.	Empty	No
<code>BACKUP_WORK_DIR</code>	Where in-app backups are assembled before download and unpacked before restore. When empty, <code>backup-work</code> inside the uploads volume, so a finished file moves into place with a rename. Under Compose any other path must be a mounted volume, since the rest of the container is read-only; never point it at <code>/tmp</code> , which is held in memory. See In-app snapshots.	Empty	No
<code>SETTINGS_ENCRYPTION_KEY</code>	Source of the key that encrypts the	Empty	No

Variable	Purpose	Default	Require
	SMTP password, LDAP bind password, SAML private key and each user's two-step verification secret in the database. When empty, the key is derived from <code>AUTH_SECRET</code> . At least 32 characters when set. Changing it (or <code>AUTH_SECRET</code> while it is empty) makes existing two-step verification enrollments unreadable; see Two-step verification.		
<code>API_RATE_LIMIT_PER_MINUTE</code>	Requests per minute each API token may make on <code>/api/v1</code> before it is refused with <code>429</code> and a <code>Retry-After</code> header. Counted per web process. <code>0</code> turns the limit off; a value that is not a whole number counts as the default.	<code>600</code>	No
<code>JOBS_DB_POOL_MAX</code>	Most database connections the job queue opens in each <code>web</code> and <code>worker</code> process, beside the	<code>10</code>	No

Variable	Purpose	Default	Require
	application's own pool (whose size is set with <code>connection_limit</code> in <code>DATABASE_URL</code>). Lower it when many processes share one database server. Any value that is not a whole number of 1 or more counts as the default.		
<code>TRUSTED_PROXY_HOPS</code>	How many reverse proxies in front of <code>web</code> append the client address to <code>X-Forwarded-For</code> . The client address is read that many entries from the right. Any value that is not a whole number of 1 or more counts as 1.	<code>1</code>	No
Database			
<code>POSTGRES_PASSWORD</code>	Password of the database owner. Used by <code>db</code> , <code>migrate</code> and the backup scripts. Keep it URL-safe.	None	Yes (Comp)

Variable	Purpose	Default	Require
<code>APP_DB_PASSWORD</code>	Password of the application role that <code>web</code> and <code>worker</code> connect as. <code>migrate</code> sets it on every start. Keep it URL-safe.	None	Yes (Comp <code>migrate</code> require character or more)
<code>POSTGRES_DB</code>	Database name.	<code>grc</code>	No
<code>POSTGRES_USER</code>	Database owner role name.	<code>grc</code>	No
<code>APP_DB_USER</code>	Application role name: lower-case letters, digits and underscores, starting with a letter or underscore, and different from <code>POSTGRES_USER</code> .	<code>grc_app</code>	No
First administrator			
<code>INITIAL_ADMIN_EMAIL</code>	Email of the administrator created when the database has no users.	Empty	Only on empty database
<code>INITIAL_ADMIN_PASSWORD</code>	Its password, 12 characters or more. Remove it after the first sign-in.	Empty	Only on empty database
HTTPS (bundled Caddy)			
<code>DOMAIN</code>	Host name Caddy serves and requests a certificate for.	Empty	With <code>-profile</code> <code>tls</code>

Variable	Purpose	Default	Require
<code>ACME_EMAIL</code>	Contact address for the certificate authority.	Empty	With <code>-profile</code> <code>tls</code>
Docker			
<code>COMPOSE_PROJECT_NAME</code>	Project name; prefixes container and volume names.	<code>grc</code>	No
<code>APP_IMAGE</code>	Tag of the application image Compose builds and runs. Leave it unset unless you need a specific name; it must differ between instances on one server.	<code>grc-platform- <COMPOSE_PROJECT_NAME>:latest</code>	No
<code>WEB_BIND</code>	Host address the <code>web</code> port is published on.	<code>127.0.0.1</code>	No
<code>WEB_PORT</code>	Host port for <code>web</code> .	<code>3000</code>	No

3.3 Set by Compose and the image

These are set for you. Do not override them unless you are changing how the image runs.

Variable	Set to	Purpose
<code>DATABASE_URL</code>	Built from the database settings above: the application role for <code>web</code> and <code>worker</code> , the owner for <code>migrate</code> .	Database connection for the application and the job queue.
<code>AUTH_URL</code>	<code>APP_URL</code>	Base address Auth.js builds its redirects from.
<code>AUTH_TRUST_HOST</code>	<code>true</code>	Tells Auth.js to trust the proxy's host header.
<code>NODE_ENV</code>	<code>production</code>	Production mode; enables the <code>AUTH_SECRET</code> check and refuses the demo seed.
<code>PORT</code> , <code>HOSTNAME</code>	<code>3000</code> , <code>0.0.0.0</code>	Where <code>web</code> listens inside its container.
<code>UPLOAD_DIR</code>	<code>/app/uploads</code>	Where attachments and in-app snapshots are stored (the <code>uploads</code> volume).
<code>NEXT_TELEMETRY_DISABLED</code> , <code>CHECKPOINT_DISABLE</code>	<code>1</code>	Turn off Next.js telemetry and Prisma's update check, so the application makes no unsolicited outbound calls.

3.4 Operator script variables

These are read by the scripts you run on the server, not by the application.

Variable	Read by	Purpose	Default
<code>BACKUP_AGE_RECIPIENTS</code>	<code>deploy/backup.sh</code>	Path to a file of age public keys. When set, both backup files are encrypted and the plain-text copies removed.	Unset (no encryption)
<code>BACKUP_AGE_IDENTITY</code>	<code>deploy/restore.sh</code>	Path to the age private key file. Required when restoring <code>.age</code> files.	Unset
<code>UPDATE_POLL_SECONDS</code>	<code>deploy/update.sh</code>	Seconds between health checks while waiting for <code>web</code> (60 checks in all).	<code>3</code>
<code>GRC_API_TOKEN</code>	<code>scripts/load-bundle.ts</code>	API token used to load a data bundle. See Loading and Importing Data.	Unset
<code>NODE_EXTRA_CA_CERTS</code>	<code>scripts/load-bundle.ts</code>	Extra CA certificates for reaching an instance with an internal certificate.	Unset

3.5 Development and demonstration only

Never set these on a production instance.

Variable	Read by	Purpose
<code>ALLOW_DEMO_SEED</code>	<code>prisma/seed.ts</code>	<code>1</code> lets the destructive development seed run when <code>NODE_ENV</code> is <code>production</code> .
<code>SEED_ADMIN_EMAIL</code> , <code>SEED_ADMIN_PASSWORD</code>	<code>prisma/seed.ts</code>	Administrator account the development seed creates.
<code>DEMO_ACCOUNTS_FILE</code>	<code>prisma/demo-seed.ts</code>	Where the demonstration seed writes its sign-in accounts (default <code>.demo-accounts</code>). The demonstration seed refuses to run on a database that holds business data.

3.6 Settings that live in the application

These are not environment variables. Set them in the application:

Setting	Address	Chapter
SMTP server, LDAP, SAML single sign-on, SSO only	<code>/settings/integrations</code>	Notifications and Email; Authentication and Single Sign-On
Session length, idle timeout, API token lifetime, attachment storage per user, required two-step verification, separation of duties	<code>/settings/security</code> (Sessions)	Authentication and Single Sign-On
SCIM provisioning	<code>/settings/provisioning</code>	Authentication and Single Sign-On
Product name, logo, color	<code>/settings/branding</code>	Platform Settings
Notification templates and the digest schedule	<code>/settings/notifications</code>	Notifications and Email
Audit log retention	<code>/settings/audit-log</code>	Audit Log and Evidence Integrity

Users, Groups and Roles

TraxrGRC controls access with three objects. A **user** is an account that can sign in. A **group** collects users and carries roles. A **role** is a permission matrix: one access level for every module and right. A user's effective rights are the union of the roles on every group they belong to, unless the account is an administrator, in which case every check passes.

Groups do a second job: records are owned by a person, a group, or both, and the *own* access level in a role means "records owned by me or by one of my groups". Designing groups is therefore designing both who holds which rights and whose records each person can reach.

This chapter covers the full lifecycle of accounts, how groups and roles combine, exactly how each access level is evaluated, the separation of duties rules, and worked examples for custom roles. It closes with a permission reference generated from the code.

4.1 Where identity is administered

The three screens sit under **Settings** and appear as cards on the Settings page:

Screen	Address	Opens with	Changes need
Users	<code>/settings/users</code>	<code>settings:read</code> and <code>users:read</code>	See Delegating user administration below
Groups	<code>/settings/groups</code>	<code>settings:read</code> and <code>users:read</code>	Administrator
Roles	<code>/settings/roles</code>	<code>settings:read</code> and <code>users:read</code>	Administrator

The page itself checks `settings:read`; the list behind it checks `users:read`. A role that grants only `users:read` therefore does not open these screens: add `settings:read` as well. None of the built-in roles grants either right, so out of the box only administrators see these three screens. People with other rights still see the Settings page, with only the cards they can open (see Map of the settings pages in Platform Settings).

Creating, editing and deleting groups and roles, granting or removing administrator rights, and changing anyone's group membership are reserved for administrator accounts whatever a role grants. These actions could hand out rights the actor does not hold, so no delegated role can perform them.

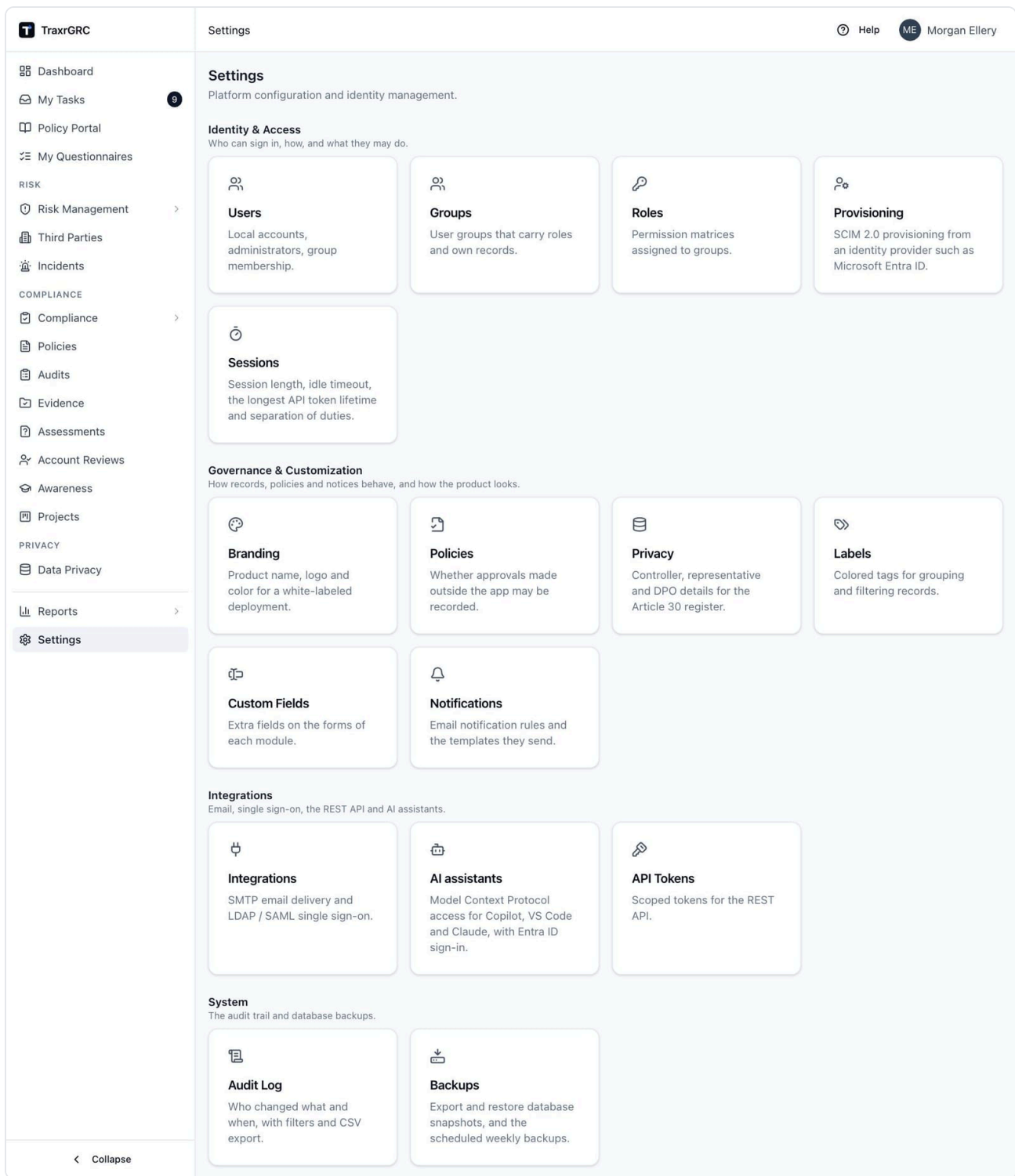


Figure 4.1. The Settings page with its cards grouped under Identity & Access, Governance & Customization, Integrations and System.

4.2 User accounts

Settings → Users lists every account, administrators first and then by name, ten to a page. Each row shows the name (with a **SCIM** badge for provisioned accounts), the email with the user's time zone below it, the groups as badges, an **Admin** badge (plus **Break-glass** where set), an **Active** or red **Inactive** badge, an **MFA** badge when the user has two-step verification on, and three actions: edit (pencil), reset password (rotate) and delete (trash).

Name	Email	Groups	Admin	Active	Actions
Administrator	install-admin@example.com	—	Admin Break-glass	Active	
Morgan Ellery	morgan.ellery@quillmont.example America/New York	—	Admin	Active	
Amara Nwosu	amara.nwosu@quillmont.example	All Staff	—	Active	
Arjun Mehta	arjun.mehta@quillmont.example America/New York	Security Engineering All Staff	—	Active	
Bianca Holm	bianca.holm@quillmont.example	All Staff	—	Active	
Chloe Fairweather	chloe.fairweather@quillmont.example Europe/Dublin	All Staff	—	Active	
Dana Whitcombe	dana.whitcombe@quillmont.example America/New York	GRC Team All Staff	—	Active	
Declan Shaw	declan.shaw@quillmont.example Europe/Dublin	All Staff	—	Active	
Elena Sokolova	elena.sokolova@quillmont.example	All Staff	—	Active	
Felix Marchetti	felix.marchetti@quillmont.example	All Staff	—	Active	

Figure 4.2. The Users list, showing each account's groups, administrator and active badges and the row actions.

4.2.1 How accounts come into existence

An account is created in one of four ways:

Source	What it creates	Password	Administrator
Bootstrap	The first administrator, from <code>INITIAL_ADMIN_EMAIL</code> and <code>INITIAL_ADMIN_PASSWORD</code> , on an installation with no users (see Installation)	Yes, at least 12 characters	Yes, and marked break-glass
Add user in Settings → Users	A local account	Yes	Optional
First LDAP or SSO sign-in	An account for an email that has no account yet, linked to the directory entry or SAML subject	No	No
SCIM provisioning	An account for each user the identity provider assigns	No	No (SCIM never grants it)

There is no invitation email and no self-registration. When you add a local user, you set the initial password and pass it to them through a channel you trust. Accounts created by LDAP, SSO or SCIM have no password and sign in only through their directory or identity provider.

4.2.2 Adding a local user

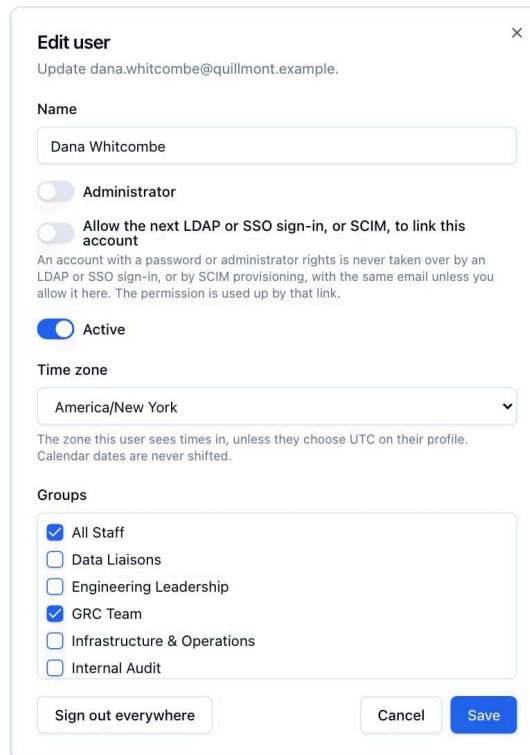
1. Open **Settings → Users** and click **Add user**. The dialog reads "Create a local account. The user signs in with email and password."
2. Complete the fields:

Field	Rules
Email	Required, a valid address, unique. Stored in lowercase. It cannot be changed later in the application.
Name	Required, up to 120 characters.
Password	8 to 128 characters with at least one letter and one digit.
Administrator	Full access to everything. Needs an administrator to set.
Break-glass (emergency password sign-in)	Shown once Administrator is on. See Break-glass administrators in Authentication and Single Sign-On.
Groups	The groups the user joins. Needs an administrator to set. SCIM-managed groups cannot be ticked here.

3. Click **Save**. The account can sign in at once. The audit log records a `users` `create` entry with the email, name, administrator and break-glass flags and the group ids. Creating an administrator also emails every active administrator a security alert ("Administrator rights granted").

4.2.3 Editing a user

Click the pencil on the row to open **Edit user**.



The 'Edit user' dialog box is a light gray window with a close button (X) in the top right corner. It contains the following elements:

- Title:** 'Edit user' with a subtitle 'Update dana.whitcombe@quillmont.example.'
- Name:** A text input field containing 'Dana Whitcombe'.
- Administrator:** A toggle switch that is currently turned off.
- Linking:** A toggle switch labeled 'Allow the next LDAP or SSO sign-in, or SCIM, to link this account' which is turned off. Below it is a small explanatory text: 'An account with a password or administrator rights is never taken over by an LDAP or SSO sign-in, or by SCIM provisioning, with the same email unless you allow it here. The permission is used up by that link.'
- Active:** A toggle switch that is currently turned on.
- Time zone:** A dropdown menu showing 'America/New York'.
- Groups:** A list of checkboxes for group membership: 'All Staff' (checked), 'Data Liaisons' (unchecked), 'Engineering Leadership' (unchecked), 'GRC Team' (checked), 'Infrastructure & Operations' (unchecked), and 'Internal Audit' (unchecked).
- Buttons:** At the bottom are three buttons: 'Sign out everywhere' (light gray), 'Cancel' (light gray), and 'Save' (blue).

Figure 4.3. The Edit user dialog with the Administrator, Active and Time zone settings and the group list.

Field	Notes
Name	Editable. For an LDAP account, the directory's name attribute overwrites it at each sign-in.
Administrator	Granting or removing it emails every administrator. You cannot remove your own.
Break-glass	Only for administrators. Removing administrator rights always clears it.
Allow the next LDAP or SSO sign-in, or SCIM, to link this account	Appears for accounts with a password, administrator rights or break-glass status. See Account linking in Authentication and Single Sign-On. The help text below it names any existing link (LDAP, SSO or SCIM provisioning).
Active	Turning it off deactivates the account. Disabled on your own account.
Time zone	The zone the user sees times in. Grayed out when the directory, identity provider or SCIM supplies it.
Groups	Membership. SCIM-managed groups are grayed out.

Reset two-step verification, shown in the dialog for anyone but yourself who has two-step verification on, turns it off for that user (see Two-step verification in Authentication and Single Sign-On).

The audit log records each save as a `users` `update` entry with the changed fields before and after; group membership appears by group name on both sides.

Sign out everywhere, at the bottom left of the dialog for anyone but yourself, ends all of that user's sessions within a minute and writes a `users` `sessions.revoke` entry. Ending an administrator's sessions needs an administrator.

Two guards protect the installation from lockout. You cannot deactivate, demote or delete your own account, and any change that would leave no active administrator is refused with "At least one active administrator account is required." When SSO-only sign-in is on, a third guard applies to the last break-glass administrator (see Authentication and Single Sign-On).

4.2.4 Deactivating and reactivating

Deactivation is the normal way to remove access. Turn **Active** off and click **Save**. The effects:

- The account can no longer sign in by any method.
- Its open sessions end within a minute.
- Its API tokens stop working at once, because every token request loads the creator's account and an inactive account resolves to no identity.

- Records it owns keep it as owner, so ownership and history stay intact.
- It stays in its groups, so reactivating restores its previous rights.

To reactivate, turn **Active** back on. Reactivating hands the account's role rights back, so a non-administrator may reactivate only an account whose rights do not exceed their own; anything more needs an administrator.

For directory accounts, deactivation in TraxrGRC is independent of the directory: an LDAP or SSO user who is inactive here is refused even if the directory accepts them. SCIM is the exception: the identity provider sets `active` and TraxrGRC follows it (see SCIM provisioning).

4.2.5 Resetting a password

Users with a local password can change it themselves on My profile (see Changing your own password under Local passwords). There is no "forgot password" link, so someone who has forgotten theirs needs an administrator (or a delegated user administrator, below) to set a new one:

1. Click the rotate icon on the user's row.
2. Enter the **New password** in the **Reset password** dialog (same rules as above) and click **Save**.

The user's open sessions end within a minute, the audit log records the reset (the password itself is never logged), and the user receives the `user.password.reset` email naming who reset it and when. Resetting an administrator's password needs an administrator. Resetting the password of an account with rights beyond your own needs an administrator, because setting someone's password lets you sign in as them.

For an account created by LDAP, SSO or SCIM, resetting gives it a local password. Do this only for a break-glass administrator who needs one; for anyone else it creates a second way in that bypasses your identity provider.

4.2.6 Deleting a user

Click the trash icon and confirm **Delete user** ("Permanently delete ...? Records they own are kept but unassigned."). Deleting:

- Clears the owner on every record the user owned. The records remain; their **Owner** field becomes empty and only people with *all*-level rights can reach them until someone assigns a new owner.
- Keeps the user's audit log entries. The entries keep the actor's email; the link to the user record is cleared.
- Is refused when the user has awareness training results or has acknowledged a policy, because those are audit evidence. The message tells you to deactivate the account instead.
- Cannot be applied to your own account (the button is disabled on your row) or to the last active administrator.

Deleting an administrator emails every administrator. Prefer deactivation in almost every case: it removes access just as completely, keeps ownership visible, and can be undone.

4.2.7 Delegating user administration

You can give a help desk or HR team limited account administration without making them administrators. Create a role with:

Module	Rights
Settings	Read (all), to open the Settings screens
Users	Read, Create, Update (all); Delete (all) only if they should delete

Holders can then add local accounts without groups, rename accounts, deactivate non-administrators, reactivate accounts whose rights are within their own, reset passwords of non-administrators whose rights are within their own, and sign other non-administrators out everywhere. They cannot change group membership, grant or remove administrator rights, allow account links, or touch an administrator's account. An administrator still adds new accounts to groups.

Be aware that `settings:read` opens every Settings screen for viewing, including Integrations (server addresses and bind DN, never the stored secrets), Sessions and Provisioning.

4.3 Groups

Settings → Groups lists each group's **Name**, **Description**, member count, **Roles** and actions. Only administrators can add, edit or delete groups.

Name	Description	Members	Roles	Actions
All Staff	Everyone at the company; the audience for...	25 members	Staff	
Data Liaisons	Business contacts who keep the records o...	3 members	Data Liaison	
Engineering Leadership	Engineering and finance leads who own en...	2 members	Risk Owner	
GRC Team	Security, risk and compliance office under ...	2 members	GRC Manager	
Infrastructure & Operations	Cloud infrastructure, networking and on-c...	3 members	Risk Owner Control Owner	
Internal Audit	Independent assurance reporting to the au...	1 member	Auditor	
People & Culture	Human resources.	1 member	—	
Privacy Office	Data Protection Officer and legal counsel.	2 members	Privacy Officer	
Procurement	Purchasing, contracts and vendor manage...	1 member	Vendor Manager	
Security Engineering	Platform security, detection and response.	2 members	Control Owner Risk Owner	

Figure 4.4. The Groups list with each group's description, member count and attached roles.

4.3.1 What a group does

A group has two independent effects:

1. **Rights.** Every member receives every role attached to the group. A user in several groups receives the union of all their groups' roles.
2. **Ownership.** Most records carry an **Owner group** besides an **Owner**. A member of the owner group counts as an owner of the record wherever a role grants *own*.

Keep the two in mind when you design groups. A group that exists only to own records can carry no roles at all (the demonstration installation's "People & Culture" group is one), and a group that exists only to grant rights may never be chosen as an owner group.

4.3.2 Creating and editing a group

1. Click **Add group**, or the pencil on a row.
2. Enter a **Name** (unique, up to 120 characters) and an optional **Description** (up to 500).
3. Tick the **Members** (listed as "Name (email)") and the **Roles**.
4. Click **Save**.

Edit group ×

Members inherit the permissions of the roles attached here.

Name

Security Engineering

Description

Platform security, detection and response.

Members

- ☐ Administrator (install-admin@example.com)
- ☐ Amara Nwosu (amara.nwosu@quillmont.example)
- ☒ Arjun Mehta (arjun.mehta@quillmont.example)
- ☐ Bianca Holm (bianca.holm@quillmont.example)
- ☐ Chloe Fairweather (chloe.fairweather@quillmont.example)
- ☐ Dana Whitcombe (dana.whitcombe@quillmont.example)

Roles

- ☐ Auditor
- ☒ Control Owner
- ☐ Data Liaison
- ☐ GRC Manager
- ☐ Privacy Officer
- ☒ Risk Owner

Cancel Save

Figure 4.5. The Edit group dialog with the name, description, member list and role list.

Every save writes a `users` `update` entry on the group with the member and role ids before and after. Users added to or removed from a group have their sessions ended within a minute so they sign in again with the new rights (your own session is left alone). Changing only the roles attached to a group does not end sessions; members' rights change at their next permission refresh, within a minute.

4.3.3 SCIM-managed groups

Groups created by SCIM provisioning carry a **SCIM** badge. Their name and members come from the identity provider, so the edit dialog grays out **Name** and **Members** and you can change only the description and the roles. Membership of these groups also cannot be changed from Settings → Users. Attaching a role to a SCIM group delegates the decision of who holds that role to whoever manages the group in your identity provider.

4.3.4 Deleting a group

Click the trash icon and confirm **Delete group**. Members keep their accounts but lose the group's roles, and their sessions end within a minute. Records whose **Owner group** was this group keep their individual owner, if any; the owner group is cleared. Check what the group owns before you delete it, because people who reached those records through *own* lose access.

4.4 Roles

Settings → Roles lists each role's **Name** (with a **System** badge for the built-in roles), **Description**, a **Permissions** summary, the **Groups** it is attached to, and actions. Only administrators can add, edit or delete roles.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

Settings > Roles

HelpME Morgan Ellery

Roles

Roles define permission matrices; groups carry them to users.

+ Add role

Name	Description	Permissions	Groups	Actions
Auditor System	Read-only across modules.	Risks (Read: all); Compliance (Read: all);...	Internal Audit	
Control Owner	Runs the controls they own: re...	Risks (Read: all); Compliance (Read: all);...	Infrastructure & Operations Security Engineering	
Data Liaison System	Answers the data mapping qu...	Data Liaison questionnaires (Read: own,...	Data Liaisons	
GRC Manager System	Full access to risk and compli...	Risks (Read: all, Create: all, Update: all, ...	GRC Team	
Privacy Officer	Runs the privacy program: rec...	Risks (Read: all, Create: all, Update: own...	Privacy Office	
Risk Owner System	Manage own risks; read share...	Risks (Read: own, Create: all, Update: o...	Infrastructure & Operations Security Engineering Engineering Leadership	
Staff	Every employee: takes awaren...	Awareness (Read: all)	All Staff	
Vendor Manager	Owns the third-party register ...	Risks (Read: all, Create: all, Update: own...	Procurement	

Figure 4.6. The Roles list with the four built-in system roles and the demonstration installation's custom roles.

4.4.1 The permission model

A role grants an access level for each pair of module and right:

- **Modules:** the 23 areas listed in the permission reference at the end of this chapter.
- **Rights:** **Read**, **Create**, **Update**, **Delete** and **Review**. Review exists only on Risks, Compliance, Policies and Data privacy; every other row shows a dash in that column.
- **Levels:** **None**, **Own** or **All**. Anything not set is None.

The editor's description reads: "Choose the access level per module and right: none, own, or all. Own is offered only where records have an owner."

Action	Rule at Own
Listing records	The list shows only records whose Owner is you or whose Owner group is one of your groups. Dashboard widgets and reports apply the same filter.
Opening, changing or deleting a record	Allowed when the record's owner is you, or its owner group is one of your groups.
Creating a record	Allowed. There is no record to own yet, so Create at Own behaves exactly like Create at All.
Working on a child record	Decided by the parent: treatments, reviews and exceptions of a risk use the risk's owner and owner group; a DPIA uses its record of processing.

Some consequences worth knowing:

- **Create at Own does not force ownership.** The person creating a record can name any owner and owner group. If they name someone else, and hold only Own on Read, they lose sight of the record as soon as it is saved. Train people to set themselves or their group as owner, or grant Read at All.
- **Update at Own lets a holder hand a record away.** Changing the owner is an update, and the new owner is not checked against the editor's groups. After saving, the editor may no longer reach the record.
- **A record with no owner and no owner group** is visible at Own to nobody. Only All-level holders and administrators see it. This is also where records land when their owner is deleted.
- **Some records have an owner but no owner group.** For these, only the personal owner counts at Own:

Module	Owner and owner group	Owner only
Risks	Risks (and their treatments, reviews and exceptions)	
Controls	Controls	
Assets	Assets	
Business units		Business units
Compliance		Compliance analyses
Policies	Policies	
Third parties	Third parties	
Incidents	Incidents	
Audits	Audit plans; audit findings (each finding has its own owner)	
Projects	Projects	
Awareness	Awareness programs	
Data privacy	Records of processing (and their DPIAs)	
Account reviews	Account review programs	
Evidence	Evidence records	
Assessments		Assessment templates and requests

- **Data Liaison questionnaires** are the exception to ownership. Own on that module means "questionnaires assigned to me", decided by the questionnaire's Data Liaison, never by an owner field. Anyone else gets "not found".
- **Own is not offered** on modules whose records have no owner: Custom fields, Reports, Dashboard, Settings, Users, Notifications, Audit log and API tokens. There it would mean the same as All. If an older role still holds Own on one of these modules, TraxrGRC treats it as None until you edit and save the role.
- **Shared configuration needs All.** Some actions configure something nobody owns: managing shared dashboard widgets, defining custom fields and labels, changing risk scales and classifications, creating compliance packages, editing notification templates, managing data

mapping questionnaire templates, and reading the audit log. These require the right at All (or an administrator) even on modules where Own exists.

4.4.4 The Review right

Module	What Review allows
Risks	Completing risk reviews; deciding risk exceptions that have no designated approver
Compliance	Deciding compliance exceptions that have no designated approver
Policies	Deciding policy exceptions that have no designated approver; recording approvals made outside the application (when that setting is on, see Platform Settings); putting a policy on the Policy Portal or taking it off (together with Update)
Data privacy	Accepting, editing or rejecting Data Liaison answers (together with Update)

Review follows the same level rules: Review at Own on Risks lets the holder complete reviews of risks they or their groups own.

4.4.5 The Approve right

The **Approve** column exists only on Data privacy. It is the Approve DPIAs right: approving or rejecting a DPIA, in the approver's own name, at Own on the DPIAs of records they or their groups own, or at All. The built-in GRC Manager role has it at All. When this right was introduced, every role (built-in or custom) holding Data privacy Review got Approve at the same level, and every API token scoped to Data privacy Review got the Approve scope, so nobody lost the ability to decide DPIAs; remove it from the roles that should not.

Approving is separate from being the Data Protection Officer. The DPO is one named user (Settings → Privacy) who gives advice before the decision and needs no Approve right to do so, only Read on the record. A DPO who also holds Approve can advise and then decide.

An exception with a **designated approver** is decided by that approver alone, who needs only Read on the parent record, or by an administrator. Review matters only for exceptions where nobody was designated.

4.4.6 The built-in roles

Four system roles are created on every installation (the bootstrap adds any that are missing at each start). They carry the **System** badge and cannot be deleted; they can be edited.

Role	Description in the product	Purpose
GRC Manager	Full access to risk and compliance modules.	The GRC team: every right on every GRC module, Review on Risks, Compliance, Policies and Data privacy, Approve DPIAs, shared dashboard management, custom fields, scheduled reports, notification rules and templates, and reading the audit log. No Settings, Users or API tokens rights; the Settings page shows them the Custom Fields, Notifications and Audit Log cards.
Risk Owner	Manage own risks; read shared catalogs.	People accountable for risks: create risks, update and review the ones they own, read everything else, and log incidents, projects, privacy records and account reviews.
Auditor	Read-only across modules.	Internal or external auditors: Read at All on every GRC module, plus the audit log.
Data Liaison	Answers the data mapping questionnaires assigned to them. Grants nothing else; give a Data Liaison other roles as well where they need more.	Business contacts who keep records of processing current.

The full matrices are in the permission reference at the end of this chapter.

Things to know before you edit a system role:

- **Renaming is safe.** The bootstrap finds system roles by a fixed internal key, so a renamed "GRC Manager" stays the built-in role and no second copy is created. If a custom role already carries the name of a built-in role that is missing, the bootstrap creates nothing and logs a warning.
- **Upgrades may extend them.** A migration in this release added Review on Compliance and Policies to "GRC Manager" where it still had Update at All on them. Custom roles are never changed by an upgrade; extend them yourself.
- **Prefer a custom role** when you need a variation. Copy the matrix into a new role with a clear name, and leave the system role as shipped.

4.4.7 Creating a custom role

1. Click **Add role**.
2. Enter a **Name** (unique, up to 120 characters) and a **Description** (up to 500) that says who should hold it.
3. Set each cell of the matrix.

4. Click **Save**.

5. Open **Settings → Groups** and attach the role to the group or groups whose members should hold it. A role attached to no group grants nothing.

Every create, update and delete of a role writes a `users` audit entry with the full matrix before and after. Deleting a role (trash icon, **Delete role**) detaches it from its groups; members lose its rights at their next permission refresh.

4.4.8 Worked example: auditor, read-only

An external audit firm needs to see the program for the audit window, and the audit log, without being able to change anything. The built-in **Auditor** role already does this. To scope it further, for example to exclude privacy data:

1. Create a role "External Auditor" with Read at All on Risks, Compliance, Policies, Third parties, Incidents, Audits, Awareness, Projects, Account reviews, Assessments, Controls, Assets, Business units, Reports, Dashboard, Custom fields and Audit log. Leave Data privacy and every other right at None.
2. Create a group "External Audit 2026" with the auditors as members and the new role attached.
3. When the engagement ends, deactivate the accounts or remove them from the group.

Read on Audit log at All puts **Settings** in their sidebar with only the **Audit Log** card, which opens `/settings/audit-log`. The Integrity and Retention cards stay hidden: verifying the chain and pruning are administrator actions. Read on Reports lets them run and download reports; without Create, Update or Delete on Reports they cannot schedule report emails.

4.4.9 Worked example: control owner

Engineers who run controls should record tests and maintenance on their own controls and see what depends on them, without editing risks or policies. The demonstration installation ships this as a custom role named "Control Owner":

Module	Read	Create	Update	Delete
Controls	All	Own	Own	None
Risks	All	None	None	None
Compliance	All	None	None	None
Policies	All	None	None	None
Audits	All	None	Own	None
Projects	All	None	Own	None
Incidents	All	All	Own	None
Assets	All	None	None	None
Business units	All	None	None	None
Dashboard	All	None	None	None
Reports	All	None	None	None

Design notes:

- Update at Own on Controls is what lets a holder record tests and maintenance on controls they own. Set each control's **Owner** to the engineer, or its **Owner group** to the team's group, for example "Security Engineering".
- Create at Own on Controls lets them add controls; remember that it behaves like All and does not force ownership.
- Read at All on Risks and Compliance lets them see which risks and requirements depend on their controls.
- Update at Own on Audits lets them work audit findings assigned to them.

Attach the role to the team's group, which should also be the owner group on the team's controls.

4.4.10 Worked example: delegating exception decisions

To let a department head decide risk exceptions for their area only:

1. Make sure the area's risks have that department's group as **Owner group**.
2. Create a role with Read at Own and Review at Own on Risks.
3. Attach it to a group containing the department head.

Exceptions on risks owned by their department, requested without a designated approver, can then be decided by them; exceptions elsewhere cannot.

4.5 Separation of duties

Several decisions require a second person by default:

Rule	Where
The person who requested a risk, compliance or policy exception cannot decide it	Exceptions on risks, compliance items and policies
The person who requested an in-app policy approval cannot approve it	Policy approvals
A Data Liaison cannot accept or reject their own questionnaire answers	Data mapping review
Nobody can name themselves as the approver of their own exception	Exception requests; always enforced

An administrator can relax the first three for teams too small to have a second person decide.

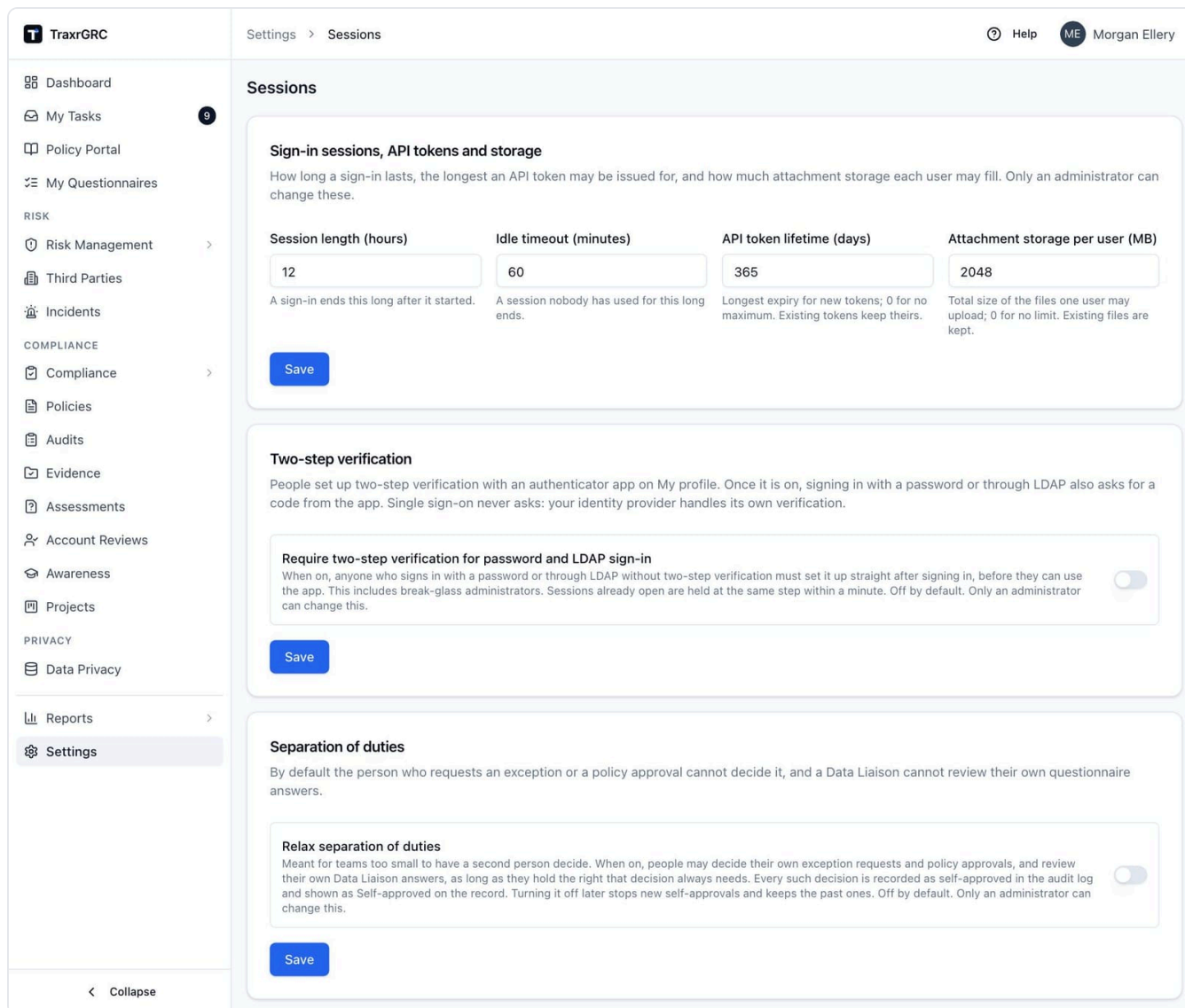


Figure 4.8. The Sessions screen with the Sign-in sessions, API tokens and storage card, the Two-step verification card and the Separation of duties card.

To relax them:

1. Open **Settings → Sessions** (address `/settings/security`).
2. In the **Separation of duties** card, turn on **Relax separation of duties**. A warning appears:
"Auditors usually expect a second person on these decisions. Every administrator will be emailed about this change."
3. Click **Save** in that card.

With the setting on, the requester may decide their own exception or approval, and a Data Liaison who also holds Review and Update on Data privacy may review their own answers. When the DPIA setting **Require a second person to approve DPIAs** (Settings → Privacy) is on, relaxing also lets a DPIA's author approve or reject it. In every case the person still needs the right the decision always needs. Every decision that succeeded only because of the setting is stored as self-approved, carries

`selfApproved: true` in its audit entry, and shows a **Self-approved** badge on the record (**Self-decided** for a rejection). Turning the setting off stops new self-approvals; past ones keep their badge.

The setting never relaxes: naming yourself as an exception's approver, the content checks that bind an approval to the version approved, the DPIA lock during DPO review, outside policy approvals, the user administration limits in this chapter, the audit log protections, and API token scope limits.

Only administrators can change the setting. Saving writes an audit entry and, when the value changes, emails every administrator ("Separation of duties setting changed").

4.6 Staff and portal-only users

Every signed-in user can open the **Policy Portal** and **My profile**, whatever their roles. An account with no roles (or only roles that grant nothing relevant) is a portal-only account, the right setup for most employees:

- After sign-in they land on the Policy Portal, because they lack Dashboard read. (A Data Liaison without Dashboard read lands on **My Questionnaires** instead.)
- The sidebar shows only the entries their rights allow: Policy Portal always, and for example **Awareness** if a role grants Awareness read.
- They read published policies and acknowledge the ones they are asked to. Acknowledgement requests reach them through group membership, so put every employee in a group such as "All Staff".
- My profile lets them set their time zone and switch off notification rules and scheduled reports addressed to them.

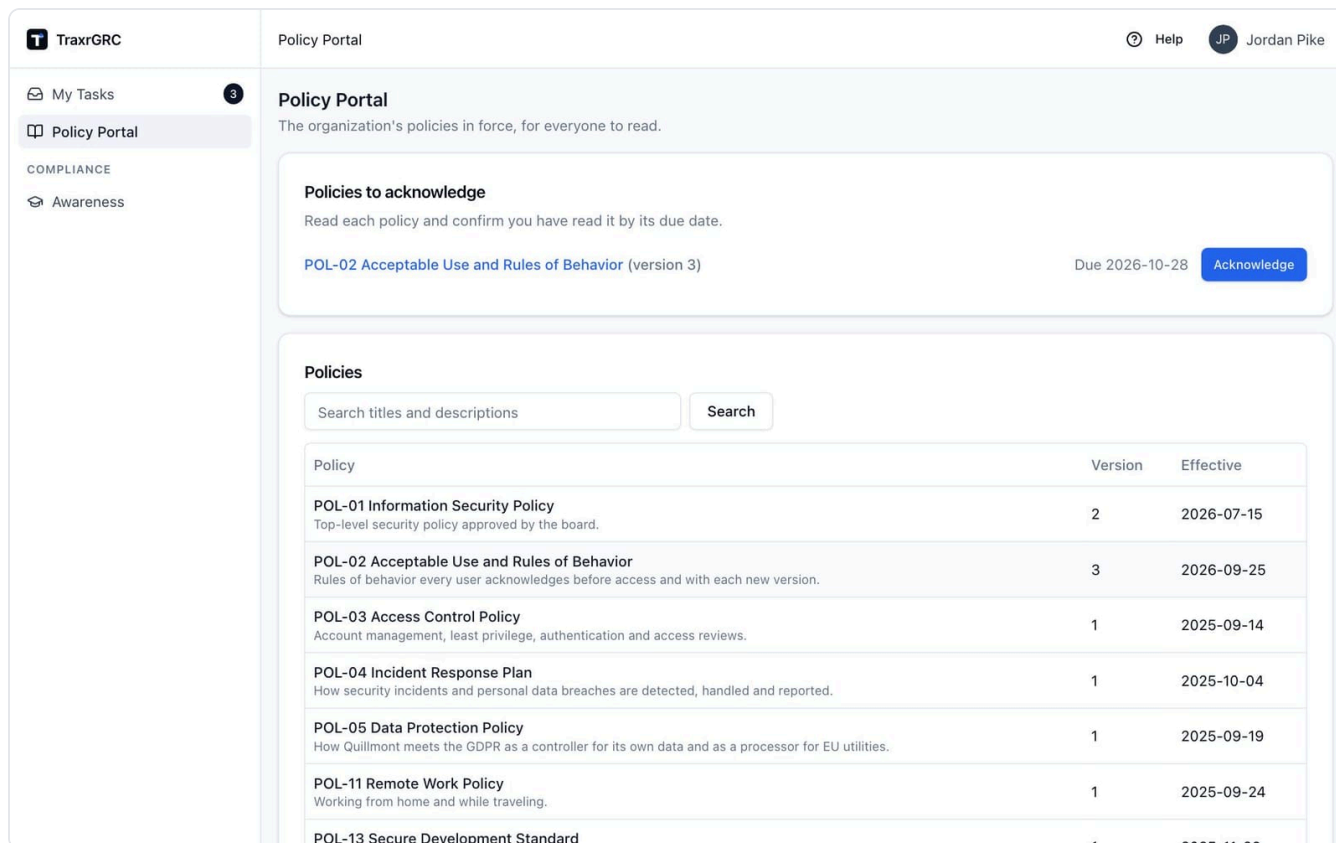


Figure 4.9. The Policy Portal as a staff member sees it, with Policies to acknowledge above the list of published policies.

The demonstration installation gives its "All Staff" group a custom "Staff" role with only Read at All on Awareness, so employees can see the training programs; the portal itself needs no role.

4.7 Effect of changes on open sessions

TraxrGRC keeps each person's rights in their session and refreshes them from the database at most once a minute while they use the application. What happens to an open session depends on the change:

Change	Effect on that user's open sessions
A role's matrix is edited	Sessions continue; the new rights apply within a minute
A role is attached to or removed from a group	Sessions continue; new rights within a minute
A role is deleted	Sessions continue; rights removed within a minute
The user is added to or removed from a group (from Users or from Groups)	Sessions end within a minute
A group the user belongs to is deleted	Sessions end within a minute
Administrator rights granted or removed	Sessions end within a minute
Account deactivated	Sessions end within a minute; API tokens stop at once
Password reset	Sessions end within a minute
Sign out everywhere	Sessions end within a minute
SSO-only turned on	Password and LDAP sessions end within a minute, except break-glass administrators
Session length changed	Applies to sessions started afterwards
Idle timeout changed	Applies to every session at its next refresh

None of these apply to your own account when you edit it yourself: your session continues and picks up the change at its next refresh.

API tokens act with the rights of the user who created them, loaded at every request, so role and group changes take effect for tokens immediately. See API Access and Integrations.

4.8 Permission reference

This section is generated from the permission engine (`src/lib/permissions/index.ts`) and the system role definitions (`prisma/reference-data.ts`). Permission names in the code use the module key shown in the first column, for example `thirdParties:update` .

4.8.1 Modules

Module key	Label in the role editor	Own offered	Review offered	What Read opens
risks	Risks	Yes	Yes	The risk register
compliance	Compliance	Yes	Yes	Compliance analyses, packages and gaps
policies	Policies	Yes	Yes	The policies list (the Policy Portal needs no right)
thirdParties	Third parties	Yes	No	Third parties
incidents	Incidents	Yes	No	Incidents
audits	Audits	Yes	No	Audit plans and findings
awareness	Awareness	Yes	No	Awareness programs
projects	Projects	Yes	No	Projects
dataPrivacy	Data privacy	Yes	Yes	Records of processing, DPIAs, transfers and the data map
dataLiaison	Data Liaison questionnaires	Yes (by assignment)	No	My Questionnaires
accountReviews	Account reviews	Yes	No	Account review programs
assessments	Assessments	Yes	No	Online assessment templates and requests
evidence	Evidence	Yes	No	Evidence records, their files and links
customFields	Custom fields	No	No	The Custom Fields screen and values on records
reports	Reports	No	No	The Report Center
controls	Controls	Yes	No	Controls
assets	Assets	Yes	No	Assets
businessUnits	Business units	Yes	No	Business units

Module key	Label in the role editor	Own offered	Review offered	What Read opens
<code>dashboard</code>	Dashboard	No	No	The Dashboard
<code>settings</code>	Settings	No	No	The Settings page and its screens
<code>users</code>	Users	No	No	The lists on Users, Groups and Roles (with <code>settings:read</code>)
<code>notifications</code>	Notifications	No	No	The Notifications screen
<code>auditLog</code>	Audit log	No	No	The Audit Log screen
<code>apiTokens</code>	API tokens	No	No	The API Tokens screen

4.8.2 Rights on configuration modules

On the record modules, Create, Update and Delete mean what they say on that module's records. On the configuration modules they mean:

Module	Read	Create	Update	Delete
Custom fields	View definitions and values	Define fields (All)	Change definitions (All)	Delete definitions (All)
Reports	Run and download reports	Schedule report emails	Change schedules	Delete schedules
Dashboard	See the Dashboard and customize your own copy of it	(unused)	Add, move and remove shared widgets (All)	(unused)
Settings	Open Settings screens	Create labels and other shared catalog entries (All)	Change branding, risk scales and other shared configuration (All)	Delete shared configuration (All)
Users	List users, groups and roles	Add local users (without groups)	Rename, deactivate, reactivate, reset passwords, sign out (non-administrators, within your own rights)	Delete non-administrator users
Notifications	View the screen	(administrator only)	Change notification settings and templates (All)	(administrator only)
Audit log	Read the audit log (All)	(unused)	(unused)	(administrator only: pruning)
API tokens	List your own tokens; All lists everyone's	Create tokens	Revoke your own; All revokes anyone's	(unused)

Integrations, Provisioning, the Sessions card, separation of duties, Policies and Privacy settings, AI assistants, Backups, audit log verification and pruning are administrator only, whatever a role grants.

4.8.3 Built-in role matrices

Levels per right: R Read, C Create, U Update, D Delete, V Review, A Approve. Cells not listed are None.

Module	GRC Manager	Risk Owner	Auditor	Data Liaison
Risks	R C U D V all	R own; C all; U own; V own	R all	
Compliance	R C U D V all	R all	R all	
Policies	R C U D V all	R all	R all	
Third parties	R C U D all	R all	R all	
Incidents	R C U D all	R all; C all; U own	R all	
Audits	R C U D all	R all	R all	
Awareness	R C U D all	R all	R all	
Projects	R C U D all	R all; C all; U own	R all	
Data privacy	R C U D V A all	R all; C all; U own	R all	
Data Liaison questionnaires				R own; U own
Account reviews	R C U D all	R all; C all; U own	R all	
Assessments	R C U D all	R all	R all	
Evidence	R C U D all	R all; C all; U own	R all	
Custom fields	R C U D all	R all	R all	
Reports	R C U D all	R all	R all	
Controls	R C U D all	R all	R all	
Assets	R C U D all	R all	R all	
Business units	R C U D all	R all	R all	
Dashboard	R C U D all	R all	R all	
Settings				
Users				
Notifications	R U all			

Module	GRC Manager	Risk Owner	Auditor	Data Liaison
Audit log	R all		R all	
API tokens				

Authentication and Single Sign-On

TraxrGRC supports three ways to sign in: local passwords, LDAP directory passwords, and SAML 2.0 single sign-on. Password and LDAP sign-in can add two-step verification with an authenticator app. SCIM 2.0 provisioning keeps accounts and groups in step with an identity provider. This chapter explains each one, how they interact, how sign-in is throttled, and how sessions end.

All of the settings described here are saved by administrators only. Every save is written to the audit log and emails every administrator a security alert. For the environment variables that affect sign-in (`APP_URL` , `AUTH_SECRET` , `TRUSTED_PROXY_HOPS`), see Configuration Reference.

5.1 Sign-in methods at a glance

Method	Enabled by	Who uses it	Account created on first use
Local password	Always available, unless SSO only is on	Accounts with a password	No
LDAP	Enable LDAP sign-in , Settings → Integrations	Directory users	Yes
SAML SSO	Enable SAML single sign-on , Settings → Integrations	Identity provider users	Yes
SCIM	Accept SCIM provisioning , Settings → Provisioning	Identity provider, not people	Yes, before first sign-in

Two ways give you multi-factor authentication. For password and LDAP sign-in, TraxrGRC has its own two-step verification with an authenticator app, which you can require for everyone (see Two-step verification below). For single sign-on, the identity provider's MFA applies; TraxrGRC never asks SSO users for a second code. To make the identity provider the only way in, turn on **SSO only**, so that password sign-in remains only for break-glass administrators.

5.2 Local passwords

5.2.1 Password policy

Passwords set in the application (new user, reset, a user's own change) must be 8 to 128 characters and contain at least one letter and one digit. The first administrator created by the bootstrap needs at

least 12 characters. There is no expiry, history or complexity rule beyond this, and no "forgot password" reset: someone who has forgotten their password needs an administrator (see Resetting a password). If your policy requires more, enforce it at your identity provider and use SSO only.

5.2.2 Changing your own password

Anyone whose account has a local password can change it on **My profile**, under **Sign-in security**, in the **Password** card: **Current password**, **New password** and **Confirm new password**, then **Change password**. Accounts that sign in only through LDAP, SSO or SCIM have no password here and do not see the card.

- The new password follows the policy above and must differ from the current one.
- Wrong current passwords are throttled like sign-ins, on a counter of their own for each account, and recorded as `users password.change.failed`. Too many gives "Too many wrong passwords. Wait 15 minutes and try again."
- A change signs out every other session of the user; the session they changed it from stays signed in.
- The audit log records `users password.change` (never the password), and the user receives the `user.password.changed` email.

5.2.3 Storage

Passwords are hashed with bcrypt at cost 12 and never stored or logged in clear. A hash made at a different cost is rehashed at cost 12 on the next successful sign-in. A sign-in attempt for an unknown email, or an account with no password, still runs a bcrypt comparison, so response time does not reveal which emails have accounts.

5.2.4 The sign-in sequence

When someone submits the sign-in form:

1. TraxrGRC checks the throttle (below) for that email and address.
2. It compares the password with the local account's hash.
3. If that fails and LDAP is enabled (and SSO only is off), the form tries the same email and password against the directory.
4. On success, the session is created and a `sign-in` audit entry is written with the method and `mfaPending`: true when the session is held for the two-step verification step or its setup, false otherwise (always false for SAML). For a held session, the `mfa.verified` entry (or `mfa.enabled` for a setup) records the completed sign-in.
5. If the account has two-step verification on, or it is required, the session is held at the two-step verification page until a code is accepted or the account is set up (see Two-step verification). A held session can open nothing in the application.

The page answers every failure with the same message, so it never confirms whether an email exists. The reason is in the audit log (see Audit Log and Evidence Integrity).

5.3 Sign-in throttling and logout

Throttling is stored in the database, so every application instance shares it. It counts failures within a 15-minute window:

Counter	Limit	Effect
One email from one address	5 failures	That address is refused for that account for 15 minutes
One address across all accounts	30 failures	That address is refused for every account for 15 minutes
One account from all addresses	10 failures	The account is guarded: any address that has failed against it in the window is refused; an address with no failures can still sign in

The account itself is never locked outright. A lock on the account would let anyone keep its owner out, a break-glass administrator above all. A successful sign-in clears the failures of that email and address pair; the account and address counters keep running until their window ends.

LDAP sign-ins are counted under the directory username (the part of the email before "@"). A directory outage does not count as a failure.

Each lock is recorded once in the audit log (`auth` `lockout`, or `account.guarded` when an account reaches ten failures) and emails every administrator, at most once an hour per lock and at most ten lockout alerts an hour in all. A break-glass account is named as such in the alert.

The address counted is the client address your reverse proxy reports. Set `TRUSTED_PROXY_HOPS` to the number of proxies in front of the application, or every client may appear as the proxy's address and share one counter (see Configuration Reference and Security Hardening). When no address can be determined at all, only the email and address pair counter applies.

There is no unlock button. Locks expire after 15 minutes. To let a locked-out user in sooner, they can sign in from another network, or an administrator can reset their password (the throttle keeps counting, so the new password works only from an address that is not locked).

5.4 Two-step verification

Two-step verification (MFA) adds a one-time code from an authenticator app to password and LDAP sign-in. Codes are standard time-based one-time passwords (TOTP, RFC 6238): six digits, a new one every 30 seconds, accepted for one step either side of the server's clock. It works with Microsoft Authenticator, Google Authenticator, 1Password and similar apps. Single sign-on never asks for a code; the identity provider handles its own verification.

5.4.1 Enrollment

Each person sets up two-step verification for themselves. The **Two-step verification** card on **My profile**, under **Sign-in security**, is shown to accounts with a local password or an LDAP link, and to everyone while LDAP sign-in is enabled.

1. Click **Set up two-step verification**.
2. In the authenticator app, add an account and scan the QR code shown. If the app cannot scan, type the setup key shown under it, or click **Open in authenticator app** on the device that has the app. The QR code is drawn on the server; the setup key is never sent to an outside image service.
3. Enter the six-digit code the app shows and click **Confirm**. The key expires if it is not confirmed within 15 minutes; **Get a new key** starts again.
4. Ten recovery codes are shown, once. Each works a single time in place of a code from the app. Save them, for example in a password manager, and click **I have saved my recovery codes**.

The card then reads "On since" the date, with how many of the 10 recovery codes are left. People cannot turn two-step verification off or get new recovery codes themselves: when a device is lost or the codes run out, an administrator resets it (below) and the person sets it up again.

5.4.2 The sign-in step

After a correct password (local or LDAP), a person with two-step verification on is taken to the **Two-step verification** page (</login/mfa>):

1. Enter the **Code from your authenticator app** and click **Verify**. Each code is accepted once; a code already used is refused.
2. Without the device, click **Use a recovery code instead** and enter one of the recovery codes. It is then spent.

Cancel and sign out abandons the sign-in. A session waiting at this step grants nothing and ends after 15 minutes.

Wrong codes are limited per account. The email and address counters of Sign-in throttling and logout apply as for passwords, and on top of them an account may make at most 10 code attempts in 15 minutes from all addresses together, counting codes from the app, recovery codes and the confirmation code during setup. The tenth wrong attempt locks that account's code step from every address until 15 minutes after the first attempt; unlike the password counters, this lock is outright. Signing in again with

the password does not lift it, and the page answers "Too many wrong codes. Wait 15 minutes and try again." A correct code resets the count. The lock is recorded as `auth mfa.locked` and emails every administrator ("Two-step verification locked after repeated wrong codes"). Since it follows a correct password, a lock the user did not cause means someone else knows their password: reset it.

5.4.3 Requiring two-step verification

Settings → Sessions has a **Two-step verification** card with one switch, **Require two-step verification for password and LDAP sign-in**, off by default. Anyone can view it with `settings:read`; only an administrator can save it.

When it is on, anyone who signs in with a password or through LDAP without two-step verification must set it up straight after signing in (**Set up now** on the two-step verification page) before they can use the application. This includes break-glass administrators. Sessions already open are held at the same step within a minute. Set it up on your own account before you turn the requirement on; the card warns you when you switch it on.

Saving writes a `settings update` entry for `security.requireMfa`, and a change emails every administrator ("Two-step verification requirement changed"). Turning the requirement off does not remove anyone's enrollment.

5.4.4 Resetting a user's two-step verification

When someone loses their device or runs out of recovery codes:

1. Open **Settings → Users** and click the pencil on their row. Users with two-step verification on show an **MFA** badge.
2. Click **Reset two-step verification**, then **Reset** in the confirmation.

Their secret and recovery codes are deleted, every session they have open is signed out, they receive the `user.mfa.reset` email, and every administrator is alerted ("Two-step verification reset for a user"). If the requirement is on, they set it up again at their next sign-in; otherwise they can set it up again from My profile. Resetting needs the same rights as resetting a password: `users:update`, and an administrator for an administrator's account or an account with rights beyond your own. Nobody can reset their own: the button is not shown on your own account, and the server refuses it with "You cannot reset your own two-step verification. Ask another administrator to reset it." Keep a second administrator able to reset yours.

5.4.5 Key rotation

Each user's two-step verification secret is stored encrypted with the settings encryption key (`SETTINGS_ENCRYPTION_KEY`, or a key derived from `AUTH_SECRET` when that is empty), and recovery codes are stored as keyed hashes (HMAC-SHA256) with a key derived from the same secret. Changing that secret makes every existing enrollment unreadable and every unused recovery code invalid: affected users can no longer pass the code step, and an administrator must reset each of them before

they can sign in. The web log shows `mfa.secret_unreadable` with the user id for each one it meets. (Recovery codes issued before keyed hashing was introduced on 2026-10-16 are stored unkeyed and keep working until used.)

5.4.6 Audit entries

Action	Module	When
<code>mfa.enroll.started</code>	<code>users</code>	Setup began and a key was issued
<code>mfa.enabled</code>	<code>users</code>	Setup was confirmed with a code
<code>mfa.enroll.failed</code>	<code>auth</code>	A wrong code during setup, or a refusal while locked (reason <code>locked: account</code>)
<code>mfa.verified</code>	<code>auth</code>	The sign-in step passed, with the method (<code>totp</code> or <code>recovery-code</code>) and, for a recovery code, how many are left
<code>mfa.failed</code>	<code>auth</code>	A wrong, reused or malformed code at sign-in, or a refusal while locked (reason <code>locked: account</code> when the account's code step is locked)
<code>mfa.locked</code>	<code>auth</code>	The account reached 10 code attempts in 15 minutes and its code step is locked
<code>mfa.reset</code>	<code>users</code>	An administrator reset a user's two-step verification
<code>password.change</code>	<code>users</code>	A user changed their own password
<code>password.change.failed</code>	<code>users</code>	A wrong current password, or a refusal while locked

Failed code attempts are recorded at most 30 a minute from one address, to keep the log from flooding; the throttle still counts every one.

5.5 LDAP

With LDAP enabled, people sign in with their directory password on the normal sign-in form; there is no separate button. Configure it in **Settings → Integrations, LDAP** card.

TraxrGRC

Dashboard
My Tasks
Policy Portal
My Questionnaires
RISK
Risk Management
Third Parties
Incidents
COMPLIANCE
Compliance
Policies
Audits
Evidence
Assessments
Account Reviews
Awareness
Projects
PRIVACY
Data Privacy
Reports
Settings

Settings > Integrations

Help Morgan Ellery

Integrations

Email delivery and single sign-on connections.

SMTP

Outbound email server used for notifications and digests.

Host

Port

587

Username

Password

From address

Use TLS (port 465)

Save

Send test email

LDAP

Directory authentication. Users are linked to local accounts on first sign-in.

Enable LDAP sign-in

Server URL

ldaps://directory.example.com:636

Bind DN (manager account)

StartTLS (required for ldaps:// URLs)

Bind password

Base DN

User filter

Mail attribute

Name attribute

cn

Time zone attribute (optional)

e.g. timezone

An attribute holding an IANA zone such as Europe/London. A valid value is applied at each sign-in and cannot be changed in the app; blank lets users choose their own.

Directory CA certificate (PEM, optional; blank uses the system trust store)

A directory entry needs the mail attribute to sign in. Sign-in never takes over an existing account that has a password or administrator rights unless you allow it for that user in Settings → Users.

Save

Test connection

SAML SSO

Single sign-on via a SAML 2.0 identity provider.

Enable SAML single sign-on

Figure 5.1. The Integrations screen with the SMTP and LDAP cards; the SAML SSO card follows below.

Field	Required	Notes
Enable LDAP sign-in		Master switch
Server URL	Yes	<code>ldaps://host:636</code> , or <code>ldap://host:389</code> with StartTLS. Anything else is refused.
StartTLS (required for ldap:// URLs)	For <code>ldap://</code>	Upgrades the connection before any bind
Bind DN (manager account)	Yes	The service account used to search
Bind password	Yes, to sign anyone in	Stored encrypted. An empty bind password is refused.
Base DN	Yes	Where the user search starts (subtree)
User filter	Yes	Must contain <code>{{username}}</code> , for example <code>(uid={{username}})</code> or <code>(sAMAccountName={{username}})</code>
Mail attribute	Yes	Default <code>mail</code>
Name attribute	Yes	Default <code>cn</code>
Time zone attribute (optional)		An attribute holding an IANA zone such as <code>America/New_York</code>
Directory CA certificate (PEM, optional; blank uses the system trust store)		For a directory certificate from an internal CA

How a sign-in works:

1. The username is the part of the typed email before "@". It is escaped and substituted into the user filter.
2. TraxrGRC binds as the manager account and searches the base DN. If the filter matches more than one entry, sign-in is refused.
3. It binds as the found entry with the typed password to prove it.
4. The entry's mail attribute becomes the account email. An entry without one cannot sign in; the typed address is never trusted as the email.
5. The account is found by its directory link, or linked or created by email (see Account linking below). The name is updated from the directory at each sign-in.

TLS 1.2 or later is required and the certificate is always verified. Connect times out after 5 seconds, each operation after 10, and the whole exchange after 20. **Test connection** binds as the manager

account and reports "Connected. Found N directory user(s)." counting entries with a mail attribute (up to 1,000).

LDAP sign-in does not read group membership from the directory. Assign groups in TraxrGRC, or use SCIM provisioning for group-driven access.

For Active Directory, a typical configuration:

Field	Value
Server URL	<code>ldaps://dc01.corp.example.com:636</code>
Bind DN (manager account)	<code>CN=svc-grc,OU=Service Accounts,DC=corp,DC=example,DC=com</code>
Base DN	<code>OU=Staff,DC=corp,DC=example,DC=com</code>
User filter	<code>(&(objectClass=user)(sAMAccountName={{username}}))</code>
Mail attribute	<code>mail</code>
Name attribute	<code>displayName</code>

This works when the part before "@" of each user's email equals their `sAMAccountName`. Where it does not, users type an address whose local part is their account name; the email recorded comes from the directory either way.

5.6 SAML single sign-on

SAML SSO sends people to your identity provider (IdP) to authenticate and accepts its signed answer. Configure it in **Settings → Integrations, SAML SSO** card.

SAML SSO
Single sign-on via a SAML 2.0 identity provider.

☐ **Enable SAML single sign-on**

☐ **SSO only: turn off password and LDAP sign-in**
Everyone signs in through the identity provider, so its MFA applies to every account. Break-glass administrators (Settings → Users) keep password sign-in for when the identity provider is down. Password sessions already open end within a minute. API tokens are unaffected.

☐ **Allow sign-in started at the identity provider**
Off: only responses to a sign-in started here are accepted, which stops someone signing a visitor in as themselves. Turn on only if people launch the app from the identity provider's portal.

IdP sign-in URL (entry point) **Issuer (SP entity ID)**

IdP entity ID (recommended: only this issuer may sign in)

Name ID format

Time zone attribute (optional)

The assertion attribute holding an IANA zone such as America/New_York. A valid value is applied at each sign-in and cannot be changed in the app; blank lets users choose their own.

IdP signing certificate (PEM)

Figure 5.2. The SAML SSO card on the Integrations screen with the switches and the identity provider fields.

5.6.1 Before you start

- `APP_URL` must be the exact external `https://` address of TraxrGRC. The service provider addresses below are built from it, and the sign-in request cookie that ties the IdP's answer to the browser is sent back on the IdP's post only when it is a secure cookie.
- Decide the **Issuer (SP entity ID)**. Any stable string works; a URL such as `https://grc.example.com` is conventional. Once your IdP trusts it, do not change it.

The addresses your IdP needs:

Item	Value
SP entity ID (Identifier, Audience)	The Issuer (SP entity ID) you enter
Assertion Consumer Service (Reply URL), HTTP-POST	<code><APP_URL>/api/auth/saml/acs</code>
Sign-on URL (where users start)	<code><APP_URL>/login</code>
SP metadata	<code><APP_URL>/api/auth/saml/metadata</code>

The metadata address works as soon as an issuer is saved, even with SAML disabled, so you can register the application with your IdP first.

5.6.2 Settings

Field	Required	Notes
Enable SAML single sign-on		Master switch. Adds Sign in with SSO to the sign-in page.
SSO only: turn off password and LDAP sign-in		See SSO only and break-glass administrators
Allow sign-in started at the identity provider		Off by default. See IdP-initiated sign-in.
IdP sign-in URL (entry point)	Yes	The IdP's SAML single sign-on URL (HTTP-Redirect binding)
Issuer (SP entity ID)	Yes	This service's entity ID; the assertion's audience must equal it
IdP entity ID (recommended: only this issuer may sign in)	Recommended	The IdP's own entity ID. When set, responses from any other issuer are refused.
Name ID format	Yes	Default <code>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</code>
Time zone attribute (optional)		The exact attribute name (it may be a URI) holding an IANA zone
IdP signing certificate (PEM)	Yes	The IdP's public signing certificate, pinned as the only trust anchor
SP private key (PEM, optional; signs AuthnRequests)		When present, authentication requests are signed. Stored encrypted.

The card's fields marked required must be filled before **Save** succeeds, even when you are saving SAML disabled. Saving records every field before and after in the audit log, except the private key (only whether it changed); the IdP certificate is recorded by its SHA-256 fingerprint, and the security alert names a changed fingerprint.

5.6.3 Attributes and account matching

TraxrGRC reads these from the assertion:

Account field	Taken from, first match wins
Email	Attribute named <code>email</code> , <code>Email</code> or <code>mail</code> ; otherwise the NameID
Name	Attribute named <code>name</code> , <code>Name</code> , <code>cn</code> or <code>displayName</code> ; otherwise the email
Time zone	The attribute named in Time zone attribute , when set
Link key	<code><issuer></code>

Attribute names are matched exactly. Claim names in URI form (for example `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`) do not match `email`; either send attributes with the short names above or make sure the NameID is the user's email address.

The account is matched first by its existing SAML link. Without one, the asserted email is matched to an account (see Account linking). The NameID must be stable for each person: if it changes, the old link no longer matches and TraxrGRC falls back to the email rule.

5.6.4 What a response must satisfy

A response is accepted only when all of these hold:

- It is signed by the pinned IdP certificate, and the assertion's signature verifies.
- Its issuer equals the **IdP entity ID**, when one is set.
- The audience equals the **Issuer (SP entity ID)**.
- The bearer subject confirmation's recipient is this instance's ACS address, and the response's destination, when present, is the same address.
- Conditions `NotOnOrAfter` is present and in the future, and the subject confirmation has an unexpired `NotOnOrAfter`.
- The assertion was issued less than 10 minutes ago (2 minutes of clock skew are tolerated).
- It answers the sign-in request this browser started, within 10 minutes, unless IdP-initiated sign-in is allowed.
- It has not been used before. Each assertion is accepted once.
- It is no larger than 64 KB (base64).

Encrypted assertions are not supported: leave assertion encryption off at the IdP. Single logout is not supported either (see Sessions and sign-out).

Every refusal sends the browser back to the sign-in page with "Single sign-on failed. Please try again or contact an administrator." The precise reason is in the audit log as an `auth` `sign-in.failed` entry with reason `saml-invalid` or `saml-replay` and a detail such as "SAML audience does not match this service provider".

5.6.5 IdP-initiated sign-in

By default TraxrGRC accepts only answers to a sign-in started from its own **Sign in with SSO** button, in the same browser, within 10 minutes. That stops someone from signing a visitor in under the attacker's identity with a response they captured. Turn on **Allow sign-in started at the identity provider** only if people launch TraxrGRC from the IdP's app portal (the My Apps tile in Entra ID, the Okta dashboard). Signed, audience, recipient, age and replay checks still apply.

5.6.6 Example: Microsoft Entra ID

1. In TraxrGRC, enter an **Issuer (SP entity ID)**, for example `https://grc.example.com`, fill the other required fields with placeholders if needed, leave **Enable SAML single sign-on** off and click **Save**.
2. In the Entra admin center, open **Enterprise applications, New application, Create your own application** (non-gallery), and choose **Single sign-on, SAML**.
3. Under **Basic SAML Configuration**, set **Identifier (Entity ID)** to the issuer, **Reply URL (Assertion Consumer Service URL)** to `https://grc.example.com/api/auth/saml/acs`, and **Sign on URL** to `https://grc.example.com/login`. You can instead upload the SP metadata.
4. Under **Attributes & Claims**, set the unique user identifier (Name ID) to `user.mail` (or `user.userprincipalname` where UPNs are email addresses) with the email address format. Add a claim named `email` with source `user.mail` and a claim named `name` with source `user.displayName`, each with the namespace field empty.
5. Under **SAML Certificates**, set the signing option to sign the SAML response and assertion, and download **Certificate (Base64)**.
6. From **Set up**, copy the **Login URL** and the **Microsoft Entra Identifier**.
7. Assign the users or groups who may sign in.
8. Back in TraxrGRC:

Field	Value
IdP sign-in URL (entry point)	The Login URL, <code>https://login.microsoftonline.com/<tenant-id>/saml2</code>
IdP entity ID	The Microsoft Entra Identifier, <code>https://sts.windows.net/<tenant-id>/</code>
Name ID format	<code>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</code>
IdP signing certificate (PEM)	The contents of the downloaded <code>.cer</code> file, including the BEGIN and END lines

9. Turn on **Enable SAML single sign-on** and click **Save**.
10. In a private browser window, click **Sign in with SSO** and complete the sign-in. Check the Audit Log for an `auth` `sign-in` entry with method `saml`.

5.6.7 Example: Okta

1. In the Okta admin console, **Applications, Create App Integration, SAML 2.0**.
2. Set **Single sign-on URL** to `https://grc.example.com/api/auth/saml/acs` (leave "Use this for Recipient URL and Destination URL" ticked) and **Audience URI (SP Entity ID)** to your issuer.
3. **Name ID format** `EmailAddress`, **Application username** `Email`.
4. Under **Attribute Statements**, add `email` (Basic) with value `user.email` and `name` (Basic) with value `user.displayName`.
5. Leave assertion encryption off. Finish, then assign people or groups.
6. From the application's **Sign On** tab, open the SAML setup instructions and copy:

Okta value	TraxrGRC field
Identity Provider Single Sign-On URL	IdP sign-in URL (entry point)
Identity Provider Issuer (<code>http://www.okta.com/<id></code>)	IdP entity ID
X.509 Certificate	IdP signing certificate (PEM)

7. Enable, save and test as for Entra ID.

5.6.8 Rotating the IdP certificate

TraxrGRC pins exactly one IdP certificate. When your IdP rolls its signing certificate, sign-ins fail from the moment it starts signing with the new one until you paste the new certificate. Plan the change: obtain the new certificate in advance, then update the **IdP signing certificate** field and save at the moment you activate it at the IdP.

Important: Keep a break-glass administrator available in case the switch goes wrong. Until the new certificate is saved, nobody can sign in through SSO.

5.7 Account linking

LDAP, SSO and SCIM all find accounts by email when no link exists yet. An outside system asserting an email must not be able to take over a local identity, so an existing account is linked automatically only when it has no password, no administrator rights and no break-glass status, or when it is already linked (for SSO: any SAML link, or provisioned by SCIM).

For any other account, the first LDAP or SSO sign-in (or SCIM request) is refused and recorded as a `link-refused` sign-in failure (`scim.link-refused` for SCIM). To allow it:

1. Open **Settings → Users** and edit the account.
2. Turn on **Allow the next LDAP or SSO sign-in, or SCIM, to link this account** and click **Save**.

The next directory sign-in, SSO sign-in or SCIM request with that email links the account, writes a `users.sso.link` entry (or `scim.link`), and turns the switch off again. Only administrators can set it.

Plan for this when you move an installation that started with local accounts to SSO: either allow the link for each account before its owner signs in with SSO, or delete accounts with no history and let SSO recreate them. Accounts you created with **Add user** all have passwords, so all of them need the link allowed.

5.8 SSO only and break-glass administrators

SSO only: turn off password and LDAP sign-in makes the identity provider the only way in, so its MFA and conditional access cover every account:

- The sign-in page shows only **Sign in with SSO**, with an **Emergency administrator sign-in** link that opens the password form.
- Password sign-in succeeds only for active administrators marked break-glass. Anyone else is refused (`sign-in.failed` with reason `sso-only`).
- LDAP sign-in is refused.
- Password and LDAP sessions already open end within a minute, except break-glass administrators'.
- API tokens are unaffected.

A break-glass administrator is an administrator account with a local password and the **Break-glass (emergency password sign-in)** switch on. The bootstrap administrator is created as one. Each break-glass password sign-in while SSO only is on is flagged in the audit log and emails every administrator ("Break-glass administrator signed in with a password").

SSO only can be saved only when SAML is enabled with an entry point and certificate, and at least one active administrator with a password is marked break-glass. The save is refused with "SSO-only cannot be turned on." and the reasons otherwise. While it is on:

- The last break-glass administrator cannot be demoted, deactivated, deleted or have the flag removed.
- SAML cannot be disabled unless SSO only is turned off in the same save.
- SCIM never changes, deactivates or deletes a break-glass administrator.

Turning SSO only off emails every administrator ("SSO-only sign-in turned off").

Warning: Set up break-glass administrators before turning on SSO only. If the identity provider fails or its certificate changes unannounced, a break-glass password is the only way back in.

Recommended setup:

1. Configure SAML and sign in once through SSO with your own account.
2. Keep two break-glass administrators with long, unique passwords held in a sealed or vaulted location, not in daily use.
3. Turn on **SSO only**.
4. Review the audit log for break-glass sign-ins as part of your access review.

5.9 SCIM provisioning

SCIM 2.0 lets an identity provider create accounts before first sign-in, keep names, emails and time zones current, deactivate leavers, and manage the membership of groups it creates. Configure it in **Settings → Provisioning** (address `/settings/provisioning`).

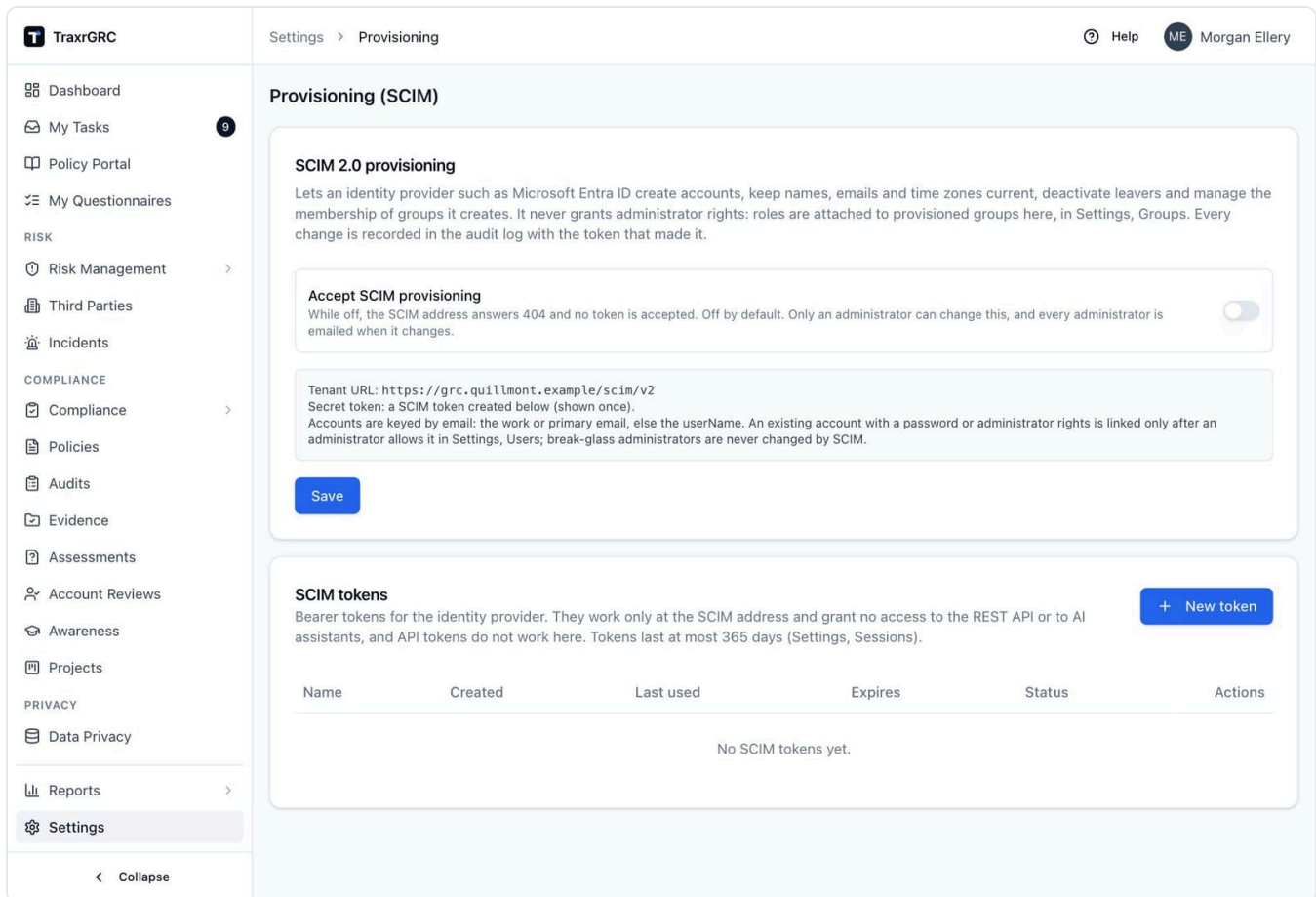


Figure 5.3. The Provisioning screen with the SCIM 2.0 provisioning switch, the tenant URL and the SCIM tokens list.

It is built to RFC 7643 and RFC 7644 and to Microsoft Entra ID's documented SCIM behavior, and tested with a scripted client replaying Entra-shaped requests. It has not been tested against a live Entra tenant; validate it in a test tenant before relying on it.

5.9.1 Endpoint and tokens

Item	Value
Tenant URL	<code><APP_URL>/scim/v2</code>
Authentication	<code>Authorization: Bearer scim_...</code> , a SCIM token
Content type	<code>application/scim+json</code> or <code>application/json</code>
While Accept SCIM provisioning is off	Every request answers 404, whatever the token

To create a token, click **New token** in the **SCIM tokens** card, enter a **Name** and an optional **Expires** date, and copy the token from the next dialog: it is shown once, and only its hash is stored. A token's

lifetime is capped by **API token lifetime (days)** in Settings → Sessions (365 by default). The list shows each token's **Name**, **Created**, **Last used**, **Expires** and **Status**, with **Rotate** (revoke and issue a replacement at once) and **Revoke**. Creating, rotating and revoking are audited (`scimToken.create` , `scimToken.rotate` , `scimToken.revoke`), and creating and rotating email every administrator.

A SCIM token works only at the SCIM address. It grants nothing on the REST API or to AI assistants, and API tokens are refused at the SCIM address.

5.9.2 Supported operations

Resource	Operations
<code>/Users</code>	<code>GET</code> list (with filter and paging), <code>POST</code> create
<code>/Users/{id}</code>	<code>GET</code> , <code>PUT</code> replace, <code>PATCH</code> , <code>DELETE</code>
<code>/Groups</code>	<code>GET</code> list, <code>POST</code> create
<code>/Groups/{id}</code>	<code>GET</code> , <code>PUT</code> , <code>PATCH</code> , <code>DELETE</code>
<code>/ServiceProviderConfig</code> , <code>/ResourceTypes</code> , <code>/Schemas</code>	<code>GET</code>

Feature	Support
Filters	<code>eq</code> only, clauses joined by <code>and</code> . Users: <code>userName</code> , <code>externalId</code> , <code>emails</code> (or <code>emails.value</code>) , <code>displayName</code> , <code>active</code> . Groups: <code>displayName</code> , <code>externalId</code> , and <code>members[value eq "<id>"]</code> .
Paging	<code>startIndex</code> and <code>count</code> , at most 100 per page
PATCH	<code>add</code> , <code>replace</code> , <code>remove</code> (operation names in any case)
Bulk, sort, ETags, password change	Not supported
Request body	At most 1 MB

User attributes read: `userName` , `emails` (primary, else work, else first), `displayName` (else `name.formatted` or the name parts), `externalId` , `active` and `timezone` . Enterprise extension attributes and other attributes are accepted and ignored.

5.9.3 What provisioning may and may not do

- An account is keyed by email: the primary or work email the provider sends, else the `userName`, which must then be an email address.
- A new email creates an account with no password and no administrator rights. The person signs in through SSO.
- An existing account is linked under the account linking rule above; otherwise the provider receives a 409 conflict and the audit log a `scim.Link-refused` entry. Accounts that were not provisioned are invisible to the provider.
- `active: false` deactivates the account and ends its sessions within a minute; `active: true` reactivates it. `DELETE` deactivates the account, removes it from SCIM groups and unlinks it from SCIM (`scim.deprovision`). Nothing is ever deleted.
- The provider creates, renames and deletes only groups it created, and manages their members (provisioned users only). A name already used by a group made in TraxrGRC is a conflict.
- Roles are never set by the provider. Attach roles to SCIM groups in Settings → Groups.
- SCIM never grants administrator rights, sets a password, changes or deactivates a break-glass administrator, reactivates an administrator, changes an administrator's email, or deactivates the last active administrator. Deactivating any other administrator is allowed and emails every administrator ("Administrator deactivated by SCIM provisioning").

Every change is audited with the actor "SCIM" and the token that made it.

5.9.4 Limits

Limit	Value	Response
Requests per token	1,200 a minute	429 with <code>Retry-After: 60</code>
Changes per token	600 a minute	429
Invalid tokens from one address	20 in 10 minutes	429 for the rest of the window

5.9.5 Setting up Microsoft Entra ID provisioning

1. In TraxrGRC, turn on **Accept SCIM provisioning**, click **Save**, then **New token**, and copy the token.
2. In Entra, open the same enterprise application you use for SAML, **Provisioning**, mode **Automatic**. Enter **Tenant URL** `https://grc.example.com/scim/v2` and **Secret Token**, then **Test Connection**.
3. User attribute mappings:

Entra attribute	SCIM attribute
<code>userPrincipalName</code> (or <code>mail</code> if UPNs are not email addresses)	<code>userName</code>
<code>Switch([IsSoftDeleted], , "False", "True", "True", "False")</code>	<code>active</code>
<code>displayName</code>	<code>displayName</code>
<code>mail</code>	<code>emails[type eq "work"].value</code>
<code>objectId</code>	<code>externalId</code>
A time zone source attribute (optional)	<code>timezone</code>

Remove or ignore the other default mappings; they are accepted and discarded. Use the same email the SAML assertion sends, so SSO finds the provisioned account.

- Group attribute mappings: `displayName` → `displayName`, `objectId` → `externalId`, `members` → `members`.
- Scope: **Sync only assigned users and groups**. Assign the users and groups and start provisioning.
- In **Settings → Groups**, attach roles to the groups that appear with the **SCIM** badge.

Provisioning and SSO work together: SCIM creates the account and its group memberships, SSO signs the person in, and because a provisioned account counts as SSO-established, the first SSO sign-in links to it without an administrator allowing it.

5.10 Sessions and sign-out

A session is an encrypted, signed cookie sealed with `AUTH_SECRET`. Its rules are set in **Settings → Sessions** (address `/settings/security`), card **Sign-in sessions, API tokens and storage**:

Field	Range	Default	Takes effect
Session length (hours)	1 to 168	12	For sessions started after the change
Idle timeout (minutes)	5 to 1440	60	For every session, at its next request
API token lifetime (days)	0 to 3650, 0 for no maximum	365	For tokens created after the change
Attachment storage per user (MB)	0 to 1048576, 0 for no limit	2048	For uploads after the change

Saving writes an audit entry and emails every administrator ("Session or API token limits changed").

A session ends at whichever comes first:

- The session length, counted from sign-in.
- The idle timeout. Activity is a page view or interaction; the browser reports activity at most once a minute.
- **Sign out** or **Sign out everywhere** in the user menu.
- An administrator's **Sign out everywhere** on the user, a password reset, a two-step verification reset, deactivation, a change of groups or administrator rights (within a minute).
- The user changing their own password on My profile, for every session but the one they changed it from.
- SSO only being turned on, for password and LDAP sessions of anyone but a break-glass administrator (within a minute).
- A change of `AUTH_SECRET`, which invalidates every session at once.

Sign-in and sign-out are recorded in the audit log (`auth` `sign-in` with the method `credentials`, `ldap` or `saml`, and `sign-out`). Signing out of GRC Platform does not sign the person out of the identity provider, and signing out of the identity provider does not end the TraxrGRC session; the session runs until one of the events above. On shared computers, keep the idle timeout short. The service provider metadata advertises no single logout service, because TraxrGRC has no single logout endpoint.

Platform Settings

This chapter describes every page where an administrator configures GRC Platform itself: the Settings area, the settings that live inside a module (risk scoring, control testing, incident classifications), and the shared vocabulary records use (labels and custom fields). Accounts, groups and roles are described in Users, Groups and Roles; sign-in, LDAP, SAML, SCIM and session length in Authentication and Single Sign-On; the Audit Log in Audit Log and Evidence Integrity; snapshots in Backup and Restore. Email and notifications have their own chapter (Notifications and Email), and API tokens and AI assistants are in API Access and Integrations.

6.1 How settings are stored and applied

Settings changed on these pages are stored in the database, in the `setting` table, and travel with every backup snapshot. The three integration secrets (SMTP password, LDAP bind password, SAML private key) are stored encrypted with the instance's settings key (see Configuration Reference). Nothing on these pages is read from environment variables, with one exception: the product name falls back to `APP_NAME` until a name is saved under Branding.

Each application process (web and worker) caches a setting for 30 seconds. A change you save takes effect at once for the page you are on and within 30 seconds everywhere else, including in the background worker that sends email. The daily sweep schedule is applied to the job queue when you save it (see Scheduler in Notifications and Email).

Every save on these pages writes an Audit Log entry with the values before and after. Saves that change security posture (integrations, session and token limits, separation of duties, SSO only, AI assistant access, SCIM) also email every administrator a security alert.

6.2 Map of the settings pages

The Settings page has a card for each of the sixteen pages below, grouped under **Identity & Access** (Users, Groups, Roles, Provisioning, Sessions), **Governance & Customization** (Branding, Policies, Privacy, Labels, Custom Fields, Notifications), **Integrations** (Integrations, AI assistants, API Tokens) and **System** (Audit Log, Backups); a group with no card for you is hidden. Each card is shown only to people the page itself admits (the **Opens with** column), so no card leads to a Forbidden page. The **Settings** entry at the foot of the sidebar is shown to anyone who sees at least one card. An administrator sees all sixteen. Of the built-in roles, a GRC Manager sees **Custom Fields**, **Notifications** and **Audit Log**; an Auditor sees **Custom Fields** and **Audit Log**; a Risk Owner sees **Custom Fields**. A Data Liaison sees no Settings entry.

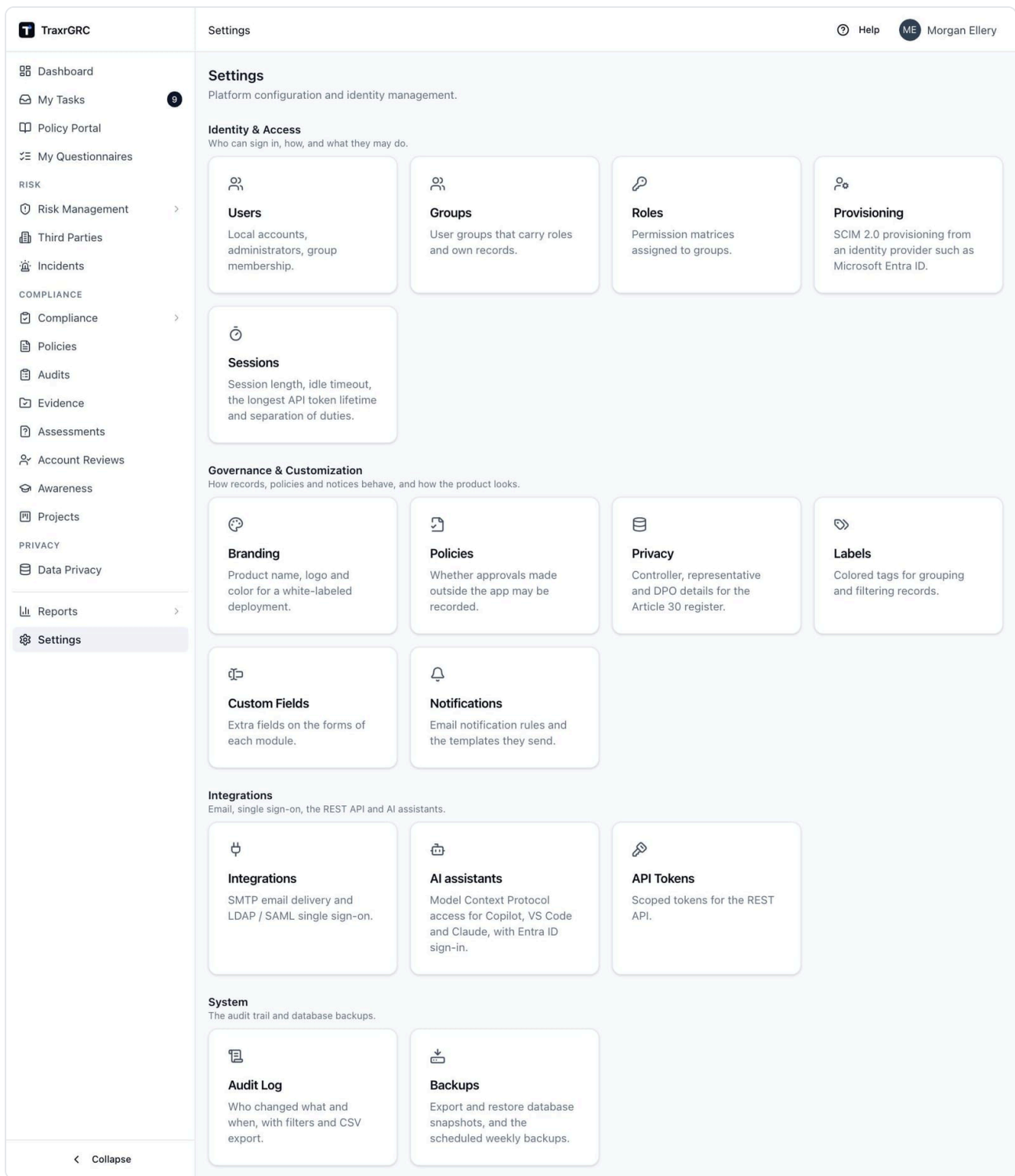


Figure 6.1. The Settings page with its cards grouped under Identity & Access, Governance & Customization, Integrations and System.

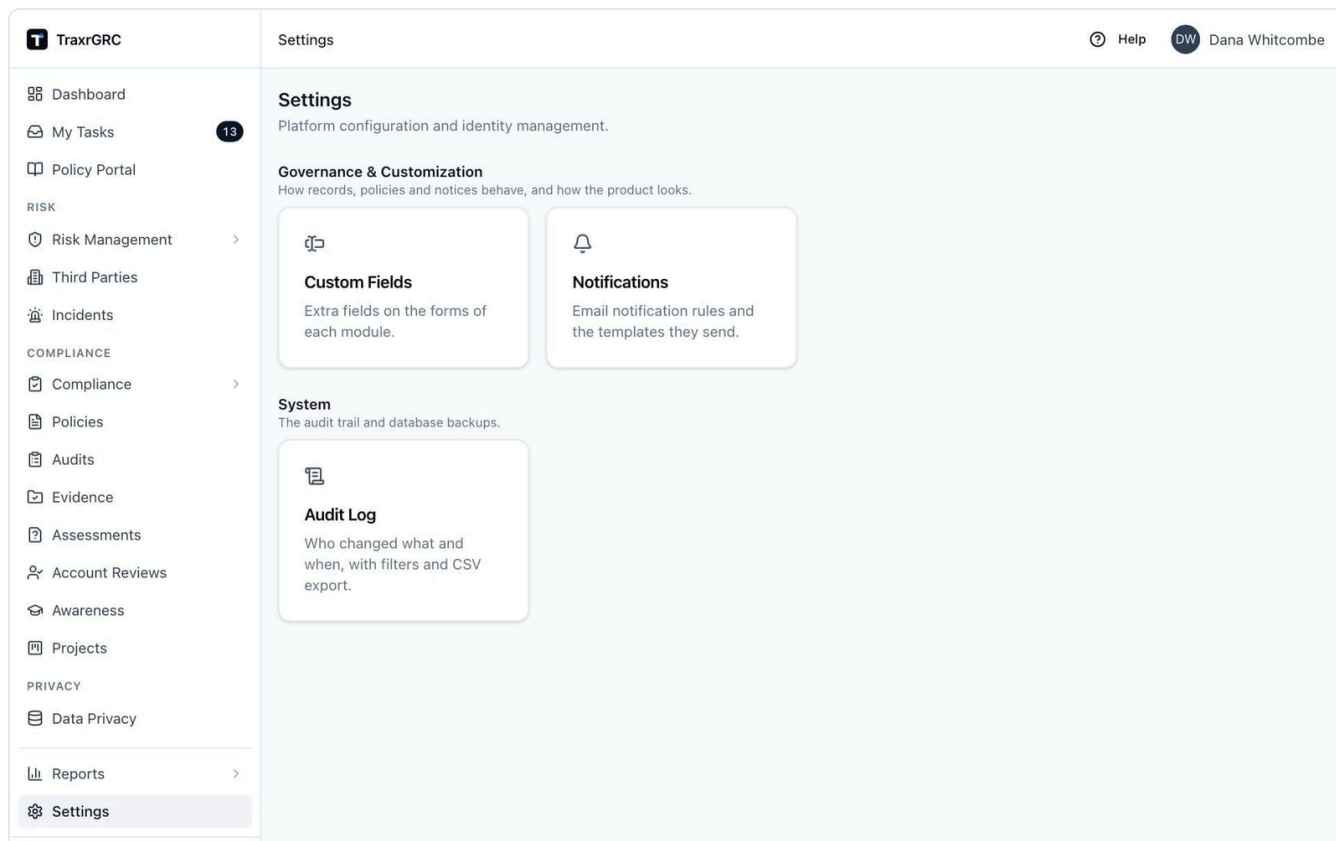


Figure 6.2. The Settings page as a GRC Manager sees it, with only the Custom Fields, Notifications and Audit Log cards.

Page	Address	Opens with	Changes need
Settings	/settings	Any one of the rights below	
Users	/settings/users	settings:read	See that chapter
Groups	/settings/groups	settings:read	See that chapter
Roles	/settings/roles	settings:read	See that chapter
Integrations	/settings/integrations	settings:read	Administrator
Provisioning	/settings/provisioning	settings:read	Administrator
Sessions	/settings/security	settings:read	Administrator
Branding	/settings/branding	settings:read	settings:update
Policies	/settings/policies	settings:read	Administrator

Page	Address	Opens with	Changes need
Privacy	<code>/settings/privacy</code>	<code>settings:read</code>	<code>settings:update</code> (header); administrator (DPIA approval)
AI assistants	<code>/settings/mcp</code>	<code>settings:read</code>	Administrator
Labels	<code>/settings/labels</code>	<code>settings:read</code>	<code>settings:create</code> , <code>settings:update</code> , <code>settings:delete</code>
Custom Fields	<code>/settings/custom-fields</code>	<code>customFields:read</code>	<code>customFields:create</code> , <code>update</code> , <code>delete</code> at all
Notifications	<code>/settings/notifications</code>	<code>notifications:read</code>	<code>notifications:update</code> ; rules: administrator
Audit Log	<code>/settings/audit-log</code>	<code>auditLog:read</code>	Integrity and Retention: administrator
API Tokens	<code>/settings/api-tokens</code>	<code>apiTokens:read</code>	<code>apiTokens:create</code> (issue); <code>apiTokens:update</code> (revoke)
Backups	<code>/settings/backups</code>	Administrator	Administrator

Three groups of settings live inside a module instead:

Settings	Where	Opens with	Changes need
Risk Settings (classifications, magnitudes, matrix, appetite bands)	<code>/risks/settings</code> , sidebar Risk Management → Risk Settings	<code>settings:read</code>	<code>settings:update</code> ; adding and removing entries also <code>settings:create</code> and <code>settings:delete</code>
Control testing settings (findings for failed audits)	Settings button on Risk Management → Controls	<code>settings:read</code> (the button is hidden without it)	<code>settings:update</code>
Incident classifications	Classifications button on Incidents	<code>incidents:update</code> at <i>all</i>	<code>incidents:update</code> at <i>all</i>

"Administrator" in these tables means an account with the **Administrator** flag. Some changes check the flag whatever a role grants, so a custom role with `settings:update` can view those pages and still cannot save them; the page shows the controls disabled or hides the **Save** button.

6.3 Branding

Settings → **Branding** white-labels the platform. Out of the box the product is called "TraxrGRC", uses the standard blue theme and a shield icon, and names no vendor anywhere. Viewing needs `settings:read` ; saving needs `settings:update` .

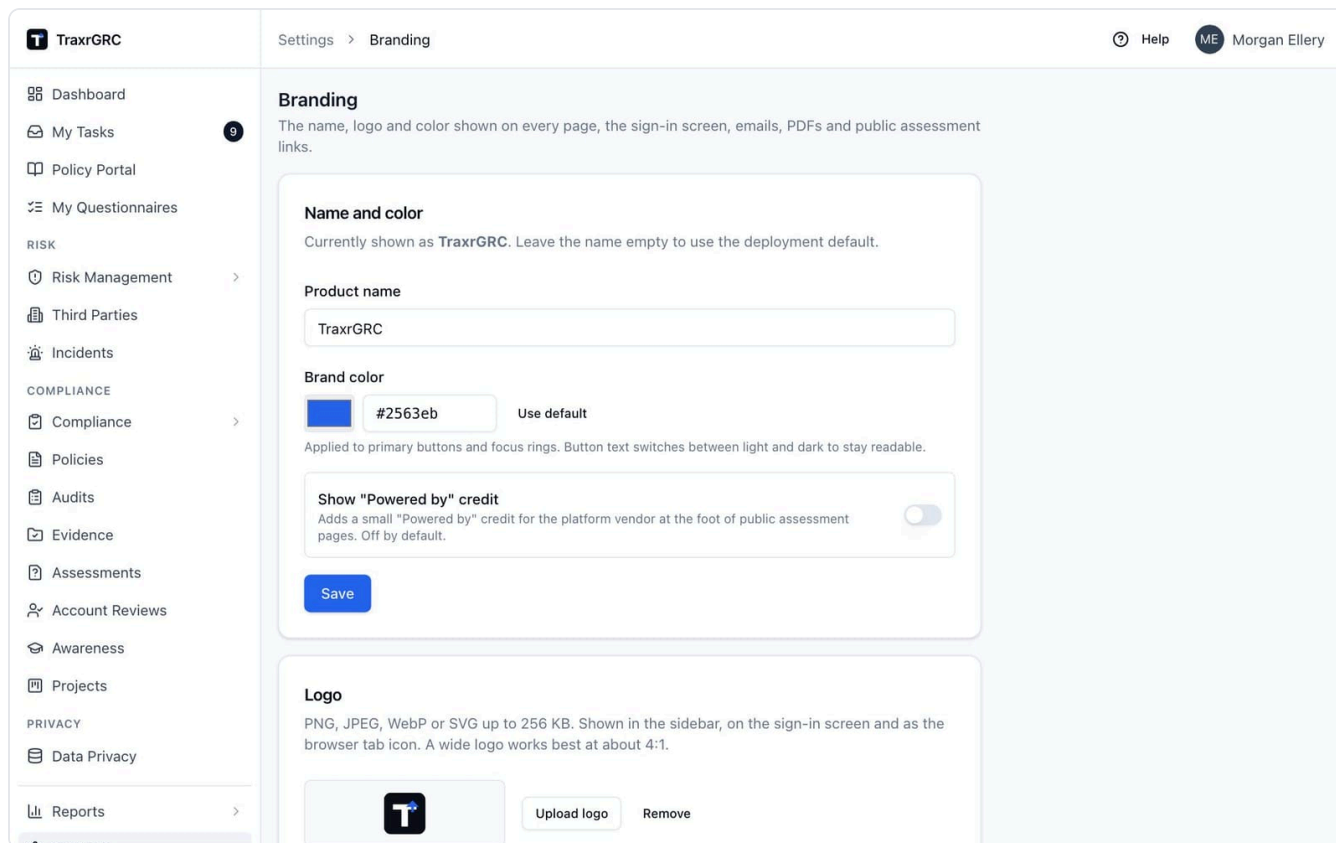


Figure 6.3. The Branding screen with the product name, brand color and logo of the demo instance.

The **Name and color** card has its own **Save** button:

Setting	Effect
Product name	Up to 60 characters. Shown in the sidebar, browser tab titles, the sign-in page, public assessment pages, PDF exports, the SMTP test email, the title of the OpenAPI document and the in-app help (the Help button), which shows the User Manual, the Quick Start Guide and, to administrators and anyone with <code>settings:read</code> , this guide, each with the product name in place of "TraxrGRC". The help is read from the guides shipped in the application image, so it matches the installed version. Email templates show it through the <code>{{app.name}}</code> variable. Leave it empty to fall back to <code>APP_NAME</code> from the deployment, and to <code>GRC Platform</code> when that is unset too. The card states the name currently in effect ("Currently shown as ...").
Brand color	A <code>#rrggbb</code> color, typed or picked. Applied to primary buttons, focus rings and links. Button text switches between light and dark to stay readable, and a color too light to read as link text on white is darkened (same hue) until it meets WCAG AA contrast. Use default clears it.
Show "Powered by" credit	Off by default. When on, public assessment pages (the ones outside contacts open from an assessment email) end with "Powered by TraxrGRC". No other page shows a vendor name.

The **Logo** card has **Upload logo** and **Remove**:

- The file may be PNG, JPEG, WebP or SVG, at most 256 KB. The type is checked from the file's content, so a renamed file is refused.
- The logo replaces the shield in the sidebar, on the sign-in page and on public assessment pages, and becomes the browser tab icon. A wide logo works best at about 4:1.
- The logo is stored in the database (and so in backups) and served from `/api/branding/logo` to anyone, since the sign-in page shows it before sign-in. An SVG is served with a policy that stops any script inside it from running.
- **Remove** returns to the shield.

Every save and every logo change is recorded in the Audit Log (module `settings`, entity `branding` or `branding.logo`).

To brand a new instance:

1. Open `/settings/branding`.
2. Enter the **Product name** your users know the service by.
3. Enter the **Brand color** as `#rrggbb`, or pick it.
4. Leave **Show "Powered by" credit** off unless your agreement asks for it.
5. Click **Save**.
6. Click **Upload logo** and choose the file.
7. Open the sign-in page in a private window to check the result. Other open sessions pick up the change within 30 seconds.

Set `APP_NAME` in the deployment as well (see Configuration Reference) when you want the right name to appear before anyone has signed in to set it, for example on the first sign-in page after installation.

6.4 Labels

Labels are short colored tags, such as `board-visibility` or `quick-win`, used to group records across modules and classifications. **Settings** → **Labels** (address `/settings/labels`) manages the list. Labels can be attached to eight record types: risks, controls, assets, third parties, incidents, policies, audit findings and projects.

- The record pages of risks, controls, third parties, incidents, policies and projects have a **Labels** picker below the description.
- Assets and audit findings have no record page of their own, so the asset list and an audit plan's **Findings** tab edit labels inline, in the **Labels** column of each row.
- Every one of these lists has a **Labels** column. Its filter finds records carrying a label whose name contains the text you type, and **Export CSV** writes the label names separated by "; ".

Name	Color	Assignments	Created	
assessor-sample	#7c3aed	4	2026-10-19 19:02 EDT	
board-visibility	#b91c1c	3	2026-10-19 19:02 EDT	
eu-data	#047857	5	2026-10-19 19:02 EDT	
fedramp-boundary	#1d4ed8	6	2026-10-19 19:02 EDT	
quick-win	#ca8a04	2	2026-10-19 19:02 EDT	

Figure 6.4. The Labels screen listing the demo's labels with their colors and assignment counts.

The page lists each label's **Name** (with its color), **Color** (hex value), **Assignments** (how many records carry it), **Created** and actions.

Action	Needs
Open the page	<code>settings:read</code>
Create label , or create one from a record's picker by typing a new name	<code>settings:create</code> at <i>all</i>
Rename or recolor (pencil)	<code>settings:update</code> at <i>all</i>
Delete (trash, then Delete label?)	<code>settings:delete</code> at <i>all</i>
Attach or detach a label on a record	Update on that record in its own module (for example <code>controls:update</code> on a control), at <i>own</i> or <i>all</i> as the role grants; without it the labels show read-only

Rules for a label:

- **Name:** lowercase letters, digits and single dashes, up to 60 characters, for example `critical-vendor`. Names are unique.
- **Color:** one of the preset swatches or any hex color (`#0ea5e9`).
- Deleting a label removes it from every record. It cannot be undone, and the Audit Log keeps the deletion.

Because creating a label from a record's picker needs `settings:create`, record owners without Settings rights can only choose labels that already exist. Create the vocabulary up front.

6.5 Custom fields

Custom fields add your own attributes to records, such as a cost center on projects or a vendor tier on third parties. **Settings → Custom Fields** (address `/settings/custom-fields`) defines them.

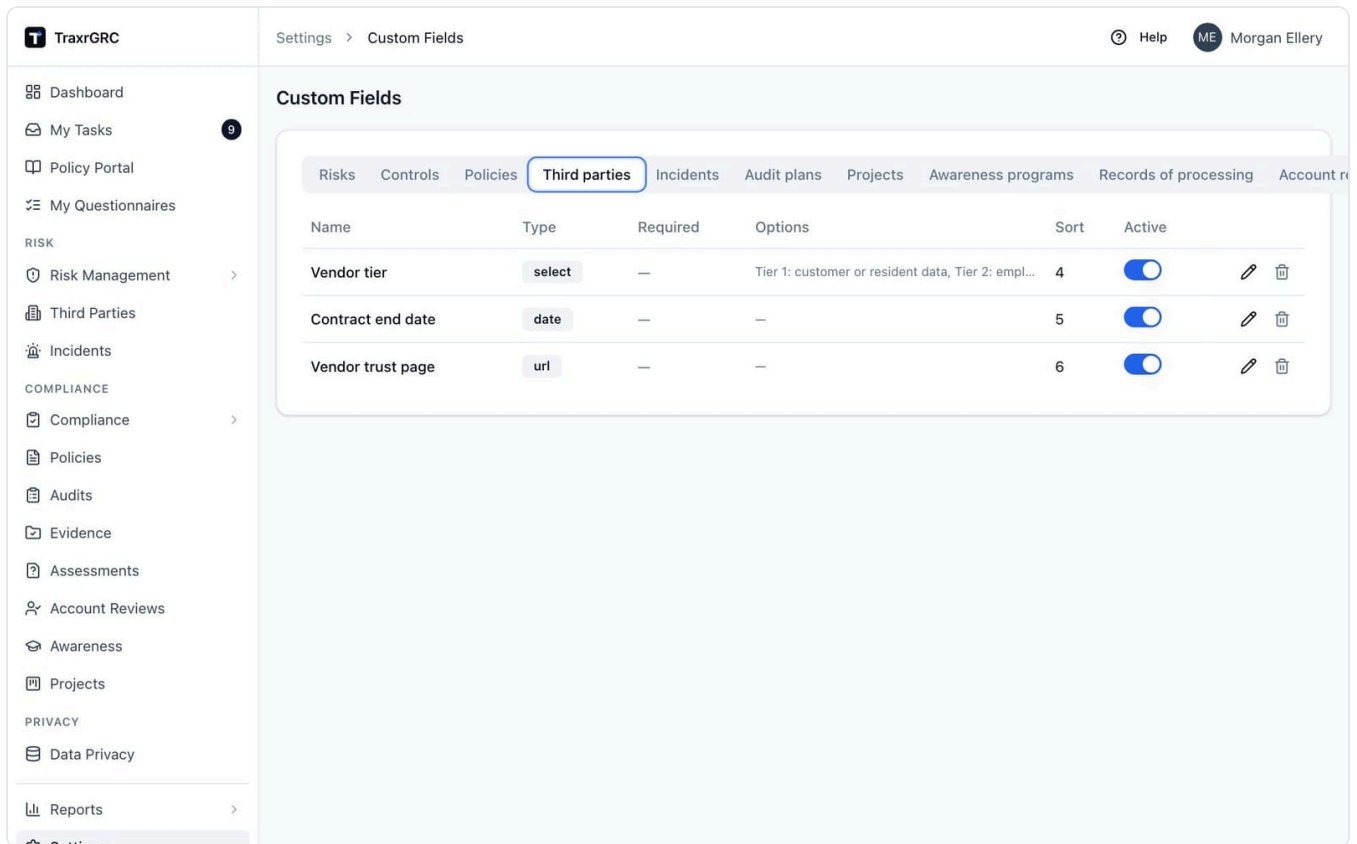


Figure 6.5. The Custom Fields screen on the Third parties tab, listing fields of several types.

6.5.1 Where custom fields apply

The page has one tab per record type. A field belongs to one record type for good; it cannot be moved to another.

Tab	Records	Module whose rights govern the values
Risks	Risks	Risks
Controls	Controls	Controls
Policies	Policies	Policies
Third parties	Third parties	Third parties
Incidents	Incidents	Incidents
Audit plans	Audit plans	Audits
Projects	Projects	Projects
Awareness programs	Awareness programs	Awareness
Records of processing	Records of processing	Data privacy
Account review programs	Account review programs	Account reviews
Compliance analyses	Compliance requirements (their analysis)	Compliance
Assessment requests	Assessment requests	Assessments

Assets, business units, audit findings and exceptions have no custom fields.

6.5.2 Field types

Type	Stored value	Limits and checks
Text	One line of text	Up to 2,000 characters.
Textarea	Multi-line text	Up to 100,000 characters.
Number	A number	Rounded to two decimal places.
Date	A calendar date	<code>YYYY-MM-DD</code> , a real date.
Select	One option	Must be one of the field's options.
Multi-select	Several options	Each must be one of the options.
Checkbox	Yes or no	
URL	A web address	<code>http://</code> or <code>https://</code> only.

Select and Multi-select fields need at least one option; options are unique, up to 200 characters each and up to 100 per field.

6.5.3 Creating and changing fields

Opening the page needs `customFields: read`. Creating, changing and deleting fields need the matching right at the *all* level. The built-in GRC Manager role holds all four; Risk Owner and Auditor can read.

1. Open the tab for the record type and click **Create field**.
2. In the **Create custom field** dialog, enter the **Name** (unique within the record type, up to 200 characters), choose the **Type** and set the **Sort order** (lower numbers first).
3. For Select and Multi-select, add the **Options** with **Add option**.
4. Tick **Required** if every record of this type must have a value.
5. Click **Create field**.

In the list, the **Active** switch hides a field from records without deleting it or its values; turn it back on to show them again. The pencil opens **Edit custom field** (save with **Save changes**); the trash deletes the field after the **Delete custom field?** confirmation, together with every value stored for it. Deleting cannot be undone.

Points to know before changing a field in use:

- **Required** is checked when someone saves the field values on a record. It does not stop a record from being created without a value, by a form, a CSV import or the API. A required checkbox must be ticked to save.

- Changing a field's **Type** does not convert values already stored. Create a new field and retire the old one instead.
- Removing an option does not change records that already hold it.
- Renaming a field changes its CSV import column (`Custom: <field name>`), so update any saved import files.

6.5.4 How custom fields appear

- On the record's page, in a **Custom Fields** tab or card listing the active fields in sort order, with required ones marked. **Edit** opens the values dialog. Viewing needs read on the record; editing needs update on it.
- In CSV imports for risks, controls, policies, third parties, incidents and projects, as extra columns headed `Custom: <field name>` (see Loading and Importing Data).
- In the REST API, at `/api/v1/custom-fields` (definitions) and `/api/v1/custom-fields/values` (values per record).

Custom fields are not shown as list columns and are not part of the Report Center reports.

6.6 Risk Settings

Risk Management → Risk Settings (address `/risks/settings`) holds the scoring engine every risk uses. It opens with `settings:read`; changing it needs `settings:update`, and adding or removing entries also needs `settings:create` or `settings:delete`, all at the *all* level. The User Manual chapter Risk Management explains how scores are calculated; this section covers what an administrator decides and what each change does to existing risks.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Risks

Controls

Assets

Business Units

Risk Settings

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

Risk Management > Risk Settings

Help Morgan Ellery

Risk Settings

Recalculate all risks

Classifications

+ New classification

Name	Slug	Order		
Asset-based	asset-based	1		
Third-party	third-party	2		
Business continuity	business-continuity	3		
Enterprise	enterprise	4		
Privacy	privacy	5		
Regulatory and contractual	regulatory-and-contractual	6		

Magnitudes

Likelihood

+ Add

Name	Value	Order		
Rare	1	1		
Unlikely	2	2		
Possible	3	3		
Likely	4	4		
Almost Certain	5	5		

Impact

+ Add

Name	Value	Order		
Insignificant	1	1		
Minor	2	2		
Moderate	3	3		
Major	4	4		
Severe	5	5		

Calculation matrix

Save matrix

L \ I	I1	I2	I3	I4	I5
L1	1	2	3	4	5
L2	2	4	6	8	10
L3	3	6	9	12	15
L4	4	8	12	16	20
L5	5	10	15	20	25

Cell colors reflect the appetite band each score falls into. Saving recalculates every risk.

Appetite bands

+ New band

Figure 6.6. The Risk Settings screen with classifications, likelihood and impact magnitudes and the color-tinted calculation matrix, with the appetite bands below.

TraxrGRC Administrator's Guide

99

6.6.1 What a new installation contains

Part	Default
Classifications	Asset-based, Third-party, Business continuity, Enterprise
Likelihood	Rare 1, Unlikely 2, Possible 3, Likely 4, Almost Certain 5
Impact	Insignificant 1, Minor 2, Moderate 3, Major 4, Severe 5
Matrix	Score = likelihood value × impact value (1 to 25)
Appetite bands	Low from 1, Medium from 6, High from 12 (over appetite), Critical from 19 (over appetite)

These are created only when the tables are empty. An upgrade never puts back a classification or band you deleted.

6.6.2 Classifications

Classifications are the tabs of the risk register. Each has a **Name**, a **Slug** (lowercase, unique; built from the name when left blank), a **Sort order** and an optional **Description**. Add one with **New classification**. A classification cannot be deleted while any risk uses it; move those risks first. The **Third-party** classification's slug (`third-party`) cannot be changed, because attaching risks to vendors depends on it; its name can. A risk moved out of it is detached from its vendor, and both the risk's and the vendor's history record that.

6.6.3 Magnitudes

The **Likelihood** and **Impact** scales, each with its own **Add** button. Each level has a **Kind**, a **Name**, a numeric **Value**, a **Sort order** and an optional **Description**. Values are unique within a scale, and risks store the value. A level's value cannot be changed, and the level cannot be deleted, while a risk or a matrix cell uses it; the refusal says how many records refer to it. Renaming a level is always allowed.

6.6.4 Calculation matrix

One score per likelihood and impact pair, labeled `L1` to `L5` and `I1` to `I5` by value. Cells are tinted with the band their score falls in. Click **Save matrix** to store all cells at once; every risk is recalculated immediately.

A pair with no cell scores 0, which no band covers, so its risks show as unscored. After adding a level, fill in its new row or column and click **Save matrix** before anyone rates a risk with it.

6.6.5 Appetite bands

Each band has a **Name**, a **Min score**, a **Color**, an **Over appetite** switch and a **Sort order**. A band covers scores from its minimum up to the next band's minimum. Bands marked over appetite drive the red banner on risks, the **Risks over appetite** dashboard widget and the `risk.appetite.breach` email. Creating, editing or deleting a band recalculates every risk.

6.6.6 Recalculate all risks

Recalculate all risks recomputes every risk's residual score from the current matrix, bands and treatments, and sends the over-appetite email for any risk that crosses into an over-appetite band. Use it after changing data outside the normal screens, for example after a database restore.

6.6.7 Changing the scale on a live register

Every save on this page rescores all risks at once, and a risk that moves into an over-appetite band emails its owner. Before changing a scale in use:

1. Agree the new scale with the risk owners and write down how old ratings map to new ones.
2. Add the new levels first, then fill the matrix and click **Save matrix**.
3. Re-rate the affected risks (by CSV import with `inherent_likelihood` and `inherent_impact`, or in the app).
4. Delete the old levels once nothing uses them.

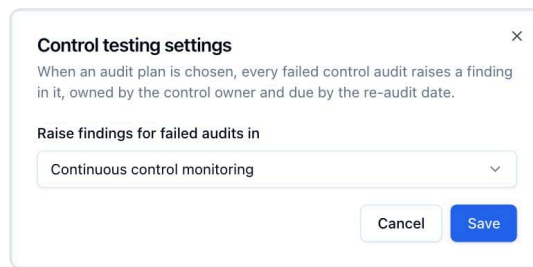
To replace levels, matrix and bands in one step, use the bundle loader's `riskScale` or `PUT /api/v1/risk-scale` (see Loading and Importing Data). That route refuses the change while any risk uses a level the new scale drops.

6.7 Control testing settings

On **Risk Management → Controls**, the **Settings** button (shown with `settings:read`) opens **Control testing settings**, a single choice:

Setting	Effect
Raise findings for failed audits in	Off: do not raise findings (the default), or an audit plan. When a plan is chosen, every failed control audit raises a finding in that plan, owned by the control owner and due by the re-audit date.

Create the audit plan first (for example "Control testing findings"); until a plan exists the dialog says so. Saving needs `settings:update`. The schedules, metrics and success criteria of individual controls are set on each control (see the User Manual, Risk Management).



Control testing settings ×

When an audit plan is chosen, every failed control audit raises a finding in it, owned by the control owner and due by the re-audit date.

Raise findings for failed audits in

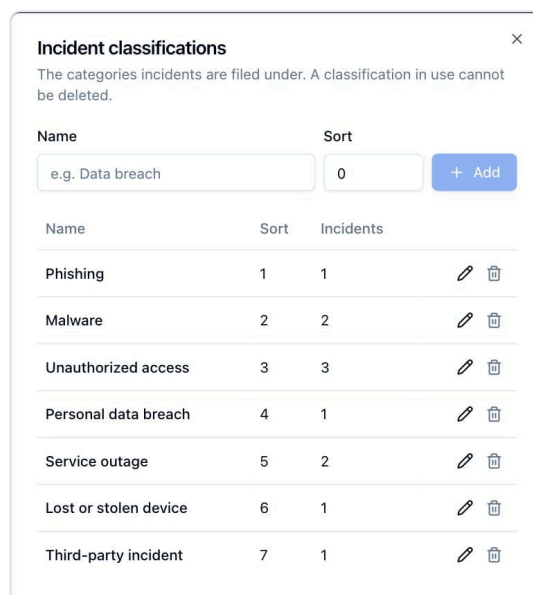
Continuous control monitoring ▼

Cancel Save

Figure 6.7. The Control testing settings dialog with the audit plan that receives findings for failed audits.

6.8 Incident classifications

Incident classifications are the categories incidents are filed under (for example Phishing or Data breach). The **Classifications** button on the Incidents page opens the **Incident classifications** dialog; it appears for people with `incidents:update` at the *all* level. Add a classification with a **Name** and a **Sort** number, rename or reorder it with the pencil, and delete it with the trash. A classification cannot be deleted while incidents use it. A new installation has none.



Incident classifications ×

The categories incidents are filed under. A classification in use cannot be deleted.

Name Sort + Add

e.g. Data breach 0

Name	Sort	Incidents	
Phishing	1	1	✎ 🗑
Malware	2	2	✎ 🗑
Unauthorized access	3	3	✎ 🗑
Personal data breach	4	1	✎ 🗑
Service outage	5	2	✎ 🗑
Lost or stolen device	6	1	✎ 🗑
Third-party incident	7	1	✎ 🗑

Figure 6.8. The Incident classifications dialog listing the demo's classifications with the fields for adding one.

6.9 Policy settings

Settings → **Policies** has one card, **Approvals made outside the app**, with one switch:

Setting	Effect
Allow recording outside approvals	Off by default. When on, people with update and Review on Policies can record an approval made outside the app (for example by a board or a controlled-document process), or upload a reissued document with one, and publish it without a second person approving in the app.

Only an administrator can change it; the change is recorded in the Audit Log. Turn it on before loading policies through the bundle loader with `effectiveDate` and `signatories` (see Loading and Importing Data).

6.10 Privacy settings

Settings → Privacy has two cards.

Records of processing header is printed at the top of the Article 30 register export. Viewing needs `settings:read` ; saving needs `settings:update` .

Field	What to enter
Controller	The organization's legal name.
Controller contact details	Postal address, email and phone.
EU or UK representative	Name and contact details, where Article 27 requires one.
Data protection officer	Name and contact details of the DPO.

Empty fields print empty.

DPIA approval has its own **Save** and is changed only by an administrator:

Setting	Stored as	Effect
Data Protection Officer	<code>privacy.dpoUserId</code>	Optional. The one user who gives DPO advice (Art. 35(2)) on DPIAs; only active users can be chosen. If that user is deactivated, no DPO is in effect (the card says so, advice stops being required, and nobody can advise) until you choose another. Independent of the DPO contact details in the header above.
DPO advice before a DPIA is decided	<code>privacy.requireDpoAdvice</code>	Required when a DPO is named (unset, the default), Required (refused unless an active DPO is named) or Optional . While required, a submitted DPIA cannot be approved or rejected until the DPO has advised; approving after a recommendation of Proceed with changes or Do not proceed needs a written reason, is shown as approved against DPO advice, and emails the DPO (<code>dataprivacy.dpia.approvedAgainstAdvice</code>). When optional, the DPO may still advise and the same reason rule applies to advice given.
Require a second person to approve DPIAs	<code>privacy.separateDpiaSignOff</code>	Off by default, so the person who wrote a DPIA may decide it (a one-person privacy office needs this). When on, whoever created the DPIA or submitted it for approval cannot approve or reject it; another holder of the Approve DPIAs right must.

Saving audits each changed setting and, when any value changes, emails every administrator ("DPIA approval settings changed").

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

< Collapse

Settings > Privacy

Help Morgan Ellery

Privacy

Records of processing header

Printed at the top of the Article 30 register export.

Controller

Quillmont Systems, Inc. and Quillmont Systems Europe Limited

The organization's legal name.

Controller contact details

Quillmont Systems, Inc., 400 Example Parkway, Columbus, OH 43215, United States; privacy@quillmont.example

Postal address, email and phone.

EU or UK representative

Quillmont Systems Europe Limited, 12 Example Quay, Dublin 2, Ireland (EU establishment; no Article 27 representative needed)

Name and contact details, where Art. 27 requires one.

Data protection officer

Sofia Brennan, Data Protection Officer, dpo@quillmont.example

Name and contact details of the DPO.

Save

DPIA approval

Approving or rejecting a DPIA needs the Data Privacy approve right, and the decision is recorded in the approver's name. The Data Protection Officer, when named, gives their advice (Art. 35(2)) before the decision.

Data Protection Officer

Sofia Brennan

Optional. Only this person can record DPO advice on a DPIA. Deactivating them leaves no DPO named until you choose another.

DPO advice before a DPIA is decided

Required when a DPO is named (default)

A DPIA awaiting approval goes to the DPO first. Approving after they advise changes or not proceeding needs a reason, and the DPO is emailed.

Require a second person to approve DPIAs

When on, whoever created a DPIA or submitted it for approval cannot approve or reject it; someone else with the approve right must. If separation of duties is relaxed (Settings, Sessions), an author may still decide, and the decision is recorded and shown as self-approved. Off by default.

Only an administrator can change these settings.

Save

Figure 6.9. The Privacy settings page with the records of processing header and the DPIA approval card.

6.11 Sessions page: token lifetime, storage and separation of duties

Settings → Sessions (address `/settings/security`) holds three cards. Anyone with `settings:read` can view them; only an administrator can save. The **Two-step verification** card is described under Two-step verification in Authentication and Single Sign-On.

The **Sign-in sessions, API tokens and storage** card:

Field	Range	Default	Effect
Session length (hours)	1 to 168	12	See Authentication and Single Sign-On.
Idle timeout (minutes)	5 to 1440	60	See Authentication and Single Sign-On.
API token lifetime (days)	0 to 3650	365	The latest expiry a new API token or SCIM token may have. A token created with no expiry date gets this lifetime. 0 removes the maximum, and a token created with no date then never expires. Existing tokens keep their expiry.
Attachment storage per user (MB)	0 to 1,048,576	2048	The total size of files one person may upload, through the app, upload links or AI assistants. 0 means no limit. Lowering it keeps existing files and refuses new uploads past the limit.

Saving the card is audited and emails every administrator.

The **Separation of duties** card has one switch, **Relax separation of duties**, off by default. By default:

- the person who requested a risk, compliance or policy exception cannot decide it;
- the person who requested an in-app policy approval cannot approve it;
- a Data Liaison cannot review their own questionnaire answers;
- when **Require a second person to approve DPIAs** is on, a DPIA's author cannot approve or reject it.

Turning the switch on lifts only the "same person" condition, for teams too small to have a second person decide. The person still needs the right the decision always needs. Every decision that succeeds only because of the switch is stored and audited as self-approved, and the record shows a **Self-approved** badge (**Self-decided** for a rejection). Turning the switch off stops new self-approvals; past ones keep their badge. Naming yourself as the approver of your own exception stays refused either way.

The card warns that auditors usually expect a second person on these decisions. Saving is audited and, when the value changes, emails every administrator. Leave it off unless the team genuinely has one person in the role, and record the decision in your control documentation.

6.12 Settings checklist for a new instance

Work through these in order after the first administrator signs in (see Installation for the steps before this):

1. **Branding**: product name, color and logo.
2. **Integrations** → **SMTP**, then **Send test email** (see Notifications and Email).
3. **Authentication**: LDAP or SAML, SCIM if used, and break-glass administrators (see Authentication and Single Sign-On).
4. **Sessions**: session and idle limits, API token lifetime, storage per user, separation of duties.
5. **Users, groups and roles** (see Users, Groups and Roles).
6. **Risk Settings**: classifications, scales, matrix and appetite.
7. **Incident classifications, Labels and Custom fields**.
8. **Policies** and **Privacy** settings.
9. **Notifications**: templates, scheduler, rules.
10. Load or import your data (see Loading and Importing Data).
11. **AI assistants** and **API tokens**, only when an integration needs them.

Notifications and Email

TraxrGRC sends plain-text email for reminders, decisions, rule matches, scheduled reports, security alerts and outside assessments. This chapter covers the mail server, the templates every email is built from, the daily sweeps that find due work, notification rules, scheduled report delivery, testing and troubleshooting.

7.1 How email is sent

1. Something in the application decides an email is due: a status change, a daily sweep finding an overdue review, a rule matching, a scheduled report coming round.
2. The application renders the template (subject and body with the placeholders filled in) and writes a row to the delivery log with status **Queued**.
3. It hands a job to the background **worker** service through the job queue in the database.
4. The worker connects to the SMTP server from **Settings → Integrations** and sends the message, then marks the row **Sent**, or **Failed** with the server's error.

The consequences for an administrator:

- **No worker, no email.** The web service only queues. If the worker container is stopped, rows stay **Queued** until it runs again.
- **Links in emails are built from** `APP_URL`. Set it to the address users reach the service at (see Configuration Reference), or every link points to the wrong host.
- **Email is plain text**, one recipient per message. There are no HTML templates.
- **Recipients are accounts**, with one exception: assessment emails (`assessment.sent`, `assessment.reminder`, `assessment.returned`) go to the outside contact named on the assessment request. No other email goes to an address that is not a user of this installation, and there are no webhooks.

7.2 Configuring SMTP

The mail server is configured on **Settings → Integrations**, in the **SMTP** card. There are no environment variables for it: the settings live in the database and are included in backups. Until it is configured, TraxrGRC tries `localhost` port 25 with the sender `grc@localhost`, which fails in a container, so every email is logged as **Failed**.

Saving and testing are administrator operations.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

Settings > Integrations

Help

Morgan Ellery

Integrations

Email delivery and single sign-on connections.

SMTP

Outbound email server used for notifications and digests.

Host

Port

587

Username

Password

From address

Use TLS (port 465)

Save

Send test email

LDAP

Directory authentication. Users are linked to local accounts on first sign-in.

Enable LDAP sign-in

Server URL

Bind DN (manager account)

ldaps://directory.example.com:636

StartTLS (required for ldaps:// URLs)

Bind password

Base DN

User filter

Mail attribute

Name attribute

(uid={{username}})

mail

cn

Time zone attribute (optional)

e.g. timezone

An attribute holding an IANA zone such as Europe/London. A valid value is applied at each sign-in and cannot be changed in the app; blank lets users choose their own.

Directory CA certificate (PEM, optional; blank uses the system trust store)

A directory entry needs the mail attribute to sign in. Sign-in never takes over an existing account that has a password or administrator rights unless you allow it for that user in Settings → Users.

Save

Test connection

SAML SSO

Single sign-on via a SAML 2.0 identity provider.

Enable SAML single sign-on

Figure 7.1. The Integrations screen with the SMTP and LDAP cards; the SAML SSO card follows below.

TraxrGRC Administrator's Guide

109

Field	What to enter
Host	The SMTP server or relay hostname. Required.
Port	587 for submission with STARTTLS, 465 for implicit TLS, 25 for an internal relay.
Username	The account to authenticate with. Leave empty for a relay that accepts the server without authentication.
Password	The account's password. Stored encrypted and never sent back to the browser; a stored one shows "Stored. Type to replace."
From address	The sender on every email, for example <code>grc@example.com</code> . Required.
Use TLS (port 465)	On for implicit TLS (port 465). Off for STARTTLS or a plain relay.

Rules the server enforces:

- **A password is never sent in clear.** With a username set and **Use TLS** off, the server must offer STARTTLS; a relay that does not is refused with an error. Use port 465 with **Use TLS**, turn on STARTTLS at the relay, or use an unauthenticated relay.
- **Changing the host or port clears a stored password** unless you type it again in the same save, so a password is never sent to a server it was not entered for.
- An unauthenticated relay is used with opportunistic STARTTLS.

To configure the mail server:

1. Open **Settings → Integrations**.
2. Fill in the **SMTP** card and click **Save**. The save is recorded in the Audit Log with the values before and after (the password only as changed or unchanged), and every administrator is sent a security alert.
3. Click **Send test email** (see below).

When an instance is restored onto a server with a different settings key, the SMTP password cannot be decrypted: it reads as empty, the server logs `settings.decrypt_failed`, and authenticated sending fails until you type the password again (see Backup and Restore).

7.3 Testing email

Send test email on the SMTP card sends a message to your own account's address, with the product name and "SMTP test" in the subject. The result appears at once: "Test email sent to your address." or "Test email failed:" with the server's reason.

The test is sent directly by the web service. To prove the whole path through the worker, check a real email in the delivery log:

1. Save the SMTP card. Saving emails every administrator a security alert.
2. Open `/settings/notifications`, tab **Delivery log**.
3. Find the `security.alert` rows for the save. **Sent** means the worker delivered them; **Failed** shows the reason in **Error**; **Queued** that stays queued means the worker is not running.

For a test instance, the deployment includes a mail catcher that accepts every message and shows it in a web page. It is meant for trying things out, never for production:

```
docker compose --profile dev up -d mailpit
```

Then set **Host** `mailpit`, **Port** `1025`, no username and **Use TLS** off, and open `http://127.0.0.1:8025` on the server to read the messages.

7.4 The Notifications page

Settings → Notifications (address `/settings/notifications`) has four tabs:

Tab	Who sees it	What it holds
Templates	<code>notifications:read</code> ; editing needs <code>notifications:update</code>	Every email's subject and body
Rules	Administrators	Notification rules
Delivery log	<code>notifications:read</code>	Every email queued; rule and scheduled report emails only for administrators
Scheduler	<code>notifications:read</code> ; saving needs <code>notifications:update</code>	When the daily sweeps run and their lead times

The built-in GRC Manager role holds read and update on Notifications. All changes are recorded in the Audit Log under the module `notifications`.

7.5 Templates

Every email TraxrGRC sends is built from a template with a fixed key. The **Templates** tab lists each one with its **Key**, **Name**, **Module**, **Subject** and an **Enabled** switch, with search and column filters. The **Module** filter offers **Risks**, **Compliance**, **Policies**, **Controls**, **Awareness**, **Data privacy**, **Assessments**, **Reports** and **Platform**.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

Settings > Notifications

Help

ME Morgan Ellery

Notifications

Email templates, notification rules, the delivery log and the daily sweeps scheduler.

Templates

Rules

Delivery log

Scheduler

Templates

Search...

Filters

Export CSV

Saved

Columns

Key ↑↓	Name ↑↓	Module ↑↓	Subject ↑↓	Enabled ↑↓
assessment.completed	Assessment completed	assessments	Assessment completed: {{assessment.title}}	☒
assessment.overdue	Assessment not answered by its due date	assessments	Assessment not answered: {{assessment.ti...	☒
assessment.reminder	Assessment reminder to a contact	assessments	Reminder: assessment due {{assessment....	☒
assessment.returned	Assessment answers returned for rework	assessments	Please review some answers: {{assessmen...	☒
assessment.sent	Assessment sent to a contact	assessments	Assessment requested: {{assessment.title}}	☒
awareness.due	Awareness program due	awareness	Awareness program due: {{program.title}}	☒
compliance.exception.expiring	Compliance exception expiring	compliance	Compliance exception expiring: {{exceptio...	☒
control.audit.due	Control audit coming due	controls	Control audit due soon: {{control.name}}	☒
control.audit.failed	Control audit failed	controls	Control audit failed: {{control.name}}	☒
control.audit.overdue	Control audit overdue	controls	Control audit overdue: {{control.name}}	☒
control.maintenance.due	Control maintenance coming due	controls	Control maintenance due soon: {{control.n...	☒

Figure 7.2. The Templates tab listing the email templates with their keys, modules, subjects and Enabled switches.

7.5.1 Template keys and their variables

The variables below are the ones each default template uses. `{{app.name}}` (the product name from Branding) works in every template.

Key	Sent when	Variables
<code>risk.review.due</code>	A risk review falls within the review lead days	<code>risk.title</code> , <code>review.dueDate</code>
<code>risk.review.overdue</code>	A risk review is past due	<code>risk.title</code> , <code>review.dueDate</code>
<code>risk.appetite.breach</code>	A risk's residual score moves into an over-appetite band	<code>risk.title</code> , <code>risk.score</code> <code>risk.band</code>
<code>risk.exception.expiring</code>	An approved risk exception expires within the exception lead days	<code>exception.title</code> , <code>exception.expiresAt</code> , <code>risk.title</code>
<code>risk.acceptance.rejected</code>	Someone rejects the exception request a review's "Accepted over appetite" result raised; sent to the reviewer	<code>risk.title</code> , <code>exception.decidedBy</code> , <code>risk.url</code>
<code>compliance.exception.expiring</code>	An approved compliance exception is expiring	<code>exception.title</code> , <code>exception.expiresAt</code> , <code>item.code</code> , <code>item.name</code>
<code>policy.review.due</code>	A policy review is coming due	<code>policy.title</code> , <code>review.dueDate</code>
<code>policy.review.overdue</code>	A policy review is past due	<code>policy.title</code> , <code>review.dueDate</code>
<code>policy.exception.expiring</code>	An approved policy exception is expiring	<code>exception.title</code> , <code>exception.expiresAt</code> , <code>policy.title</code>
<code>policy.acknowledgement.requested</code>	People are asked to acknowledge a policy	<code>policy.title</code> , <code>policy.version</code> , <code>acknowledgement.dueDate</code> <code>acknowledgement.url</code>
<code>policy.acknowledgement.due</code>	An acknowledgement is coming due	same as above

Key	Sent when	Variables
<code>policy.acknowledgement.overdue</code>	An acknowledgement is overdue	same as above
<code>awareness.due</code>	An awareness program is due and not completed	<code>program.title</code> , <code>program.dueDate</code>
<code>control.audit.due</code>	A control audit is coming due	<code>control.name</code> , <code>test.dueDate</code> , <code>control.successCriteria</code>
<code>control.audit.overdue</code>	A control audit is overdue	<code>control.name</code> , <code>test.dueDate</code> , <code>control.successCriteria</code>
<code>control.audit.failed</code>	A control audit is recorded as failed	<code>control.name</code> , <code>test.completedAt</code> , <code>test.notes</code>
<code>control.maintenance.due</code>	Control maintenance is coming due	<code>control.name</code> , <code>test.dueDate</code> , <code>control.maintenanceTask</code>
<code>control.maintenance.overdue</code>	Control maintenance is overdue	<code>control.name</code> , <code>test.dueDate</code> , <code>control.maintenanceTask</code>
<code>dataprivacy.review.due</code>	A record of processing review is coming due	<code>record.name</code> , <code>review.dueDate</code>
<code>dataprivacy.review.overdue</code>	A record of processing review is overdue	<code>record.name</code> , <code>review.dueDate</code>
<code>dataprivacy.dpa.expiring</code>	A third party's data processing agreement is expiring	<code>thirdParty.name</code> , <code>dpa.expiresAt</code>
<code>dataprivacy.dpa.expired</code>	A data processing agreement has expired	<code>thirdParty.name</code> , <code>dpa.expiresAt</code>

Key	Sent when	Variables
<code>evidence.expiring</code>	Evidence is within 30 days of its valid-until date (to its owner and, for a third-party report, the third party's owner; one email if they are the same person)	<code>evidence.title</code> , <code>evidence.type</code> , <code>evidence.thirdParty</code> (" , name" or empty), <code>evidence.validUntil</code> , <code>evidence.url</code>
<code>evidence.expired</code>	Evidence has passed its valid-until date (to the same people as <code>evidence.expiring</code>)	as <code>evidence.expiring</code>
<code>dataprivacy.tia.due</code>	A transfer impact assessment review is coming due	<code>thirdParty.name</code> , <code>tia.country</code> , <code>tia.reviewDue</code>
<code>dataprivacy.tia.overdue</code>	A transfer impact assessment review is overdue	<code>thirdParty.name</code> , <code>tia.country</code> , <code>tia.reviewDue</code>
<code>dataprivacy.questionnaire.sent</code>	A data mapping questionnaire is sent to a Data Liaison	<code>request.subject</code> , <code>request.requester</code> , <code>request.template</code> , <code>request.dueDate</code> , <code>request.message</code> , <code>request.url</code>
<code>dataprivacy.questionnaire.submitted</code>	A Data Liaison submits answers	<code>request.subject</code> , <code>request.respondent</code> , <code>request.template</code> , <code>request.url</code>
<code>dataprivacy.dpia.approvedAgainstAdvice</code>	A DPIA is approved after the DPO advised changes or not proceeding (to the DPO)	<code>dpia.title</code> , <code>dpia.record</code> , <code>dpia.approver</code> , <code>dpia.recommendation</code> , <code>dpia.reason</code> , <code>dpia.url</code>
<code>dataprivacy.attestation.sent</code>	The annual check of a record goes to its Data Liaison	<code>request.subject</code> , <code>request.dueDate</code> , <code>request.url</code>

Key	Sent when	Variables
<code>dataprivacy.questionnaire.due</code>	A questionnaire is coming due	<code>request.subject</code> , <code>request.dueDate</code> , <code>request.url</code>
<code>dataprivacy.questionnaire.overdue</code>	A questionnaire is overdue	<code>request.subject</code> , <code>request.dueDate</code> , <code>request.url</code>
<code>assessment.sent</code>	An assessment is sent to an outside contact	<code>assessment.title</code> , <code>assessment.contactName</code> , <code>assessment.dueNote</code> , <code>assessment.url</code>
<code>assessment.reminder</code>	An unanswered assessment is coming due	<code>assessment.title</code> , <code>assessment.contactName</code> , <code>assessment.dueDate</code> , <code>assessment.url</code>
<code>assessment.returned</code>	Answers are returned to the contact for rework	<code>assessment.title</code> , <code>assessment.contactName</code> , <code>assessment.flaggedCount</code> , <code>assessment.dueNote</code> , <code>assessment.url</code>
<code>assessment.overdue</code>	An assessment was not answered by its due date	<code>assessment.title</code> , <code>assessment.contact</code> , <code>assessment.dueDate</code> , <code>assessment.url</code>
<code>assessment.completed</code>	An assessment is completed	<code>assessment.title</code> , <code>assessment.contact</code> , <code>assessment.score</code>
<code>report.scheduled</code>	A scheduled report is sent	<code>schedule.name</code> , <code>report.title</code> , <code>report.generatedAt</code> , <code>attachment.note</code> , <code>report.url</code> , <code>profile.ur</code>
<code>notification.rule</code>	A notification rule matches	<code>event.summary</code> , <code>event.detail</code> , <code>record.label</code> , <code>record.title</code> ,

Key	Sent when	Variables
		<code>record.url</code> , <code>rule.name</code> , <code>profile.url</code>
<code>digest.daily</code>	The daily digest, when turned on	<code>digest.date</code> , <code>digest.body</code>
<code>security.alert</code>	A security event (see Audit Log and Evidence Integrity)	<code>alert.title</code> , <code>alert.detail</code> , <code>alert.actor</code> , <code>alert.ip</code> , <code>alert.time</code> , <code>alert.url</code>
<code>user.password.reset</code>	An administrator resets a user's password	<code>reset.by</code> , <code>reset.time</code>
<code>user.password.changed</code>	A user changes their own password on My profile	<code>change.time</code>
<code>user.mfa.reset</code>	An administrator resets a user's two- step verification	<code>reset.by</code> , <code>reset.time</code>

Dates in reminder variables are written `yyyy-MM-dd`. `alert.time`, `reset.time` and `change.time` are shown in the recipient's time zone. The `digest.daily` subject defaults to "{{app.name}} daily digest: {{digest.date}}".

7.5.2 Editing a template

1. Click the template's key. The edit dialog opens.
2. Change the **Name** (up to 120 characters), **Subject** (up to 300) or **Body**. Placeholders are written `{{variable.name}}`.
3. Check the **Available vars** panel beside the fields. It lists each variable the template receives, with its meaning, plus `app.name`.
4. Click **Save template**. The next email queued from that template uses the new text; emails already queued keep the old text.

Figure 7.3. The Edit template dialog for risk.appetite.breach with the subject, body and the Available vars panel.

Things to know:

- A placeholder the template does not receive renders as nothing; there is no error. Send yourself a test (for example with **Run sweeps now** on an instance holding test data) after editing.
- There is no "reset to default". Copy the original text somewhere before changing it. The defaults are in `prisma/reference-data.ts` in the source.
- Upgrades add new templates and never overwrite ones you edited.
- Keep the security wording in `notification.rule` and `report.scheduled` that tells people why they receive the email and links to `profile.url`, where they can opt out.
- Template edits and switches are recorded in the Audit Log with the text before and after.

7.5.3 Turning a template off

The **Enabled** switch stops a template without editing it. Nothing is sent from a disabled template: fixed notifications are logged as **Skipped** with the reason `template-disabled`, and rule and scheduled report emails as **Suppressed**. Disabling `security.alert` stops security alerts to administrators; leave it on.

7.6 Scheduler

The **Scheduler** tab holds the **Daily sweeps scheduler** card. Once a day the worker scans reviews, exceptions, control tests, awareness programs, data processing agreements, transfer impact assessments and questionnaires and queues the reminders above. The same run handles deadline notification rules, assessment reminders and repeats, and policy acknowledgement reminders.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

Settings > Notifications

Help

ME Morgan Ellery

Notifications

Email templates, notification rules, the delivery log and the daily sweeps scheduler.

Templates

Rules

Delivery log

Scheduler

Daily sweeps scheduler

The background worker scans reviews and exceptions on a cron schedule and queues notification emails. Sweeps skip items already notified in the last 20 hours, so overlapping runs never double-send.

Daily digest

When on, each recipient gets one summary email instead of individual notifications.

Cron schedule (UTC)

0 7 * * *

5 fields: minute hour day month weekday (UTC).

Review lead days

7

Reviews due within this many days.

Exception lead days

14

Exceptions expiring within this many days.

Organization time zone

America/New York

Decides which day it is for due dates, overdue items and reminders, for everyone.

Save settings

Run sweeps now

Figure 7.4. The Scheduler tab with the daily digest switch, cron schedule and lead days.

Setting	Default	What it means
Daily digest	Off	When on, each person gets one <code>digest.daily</code> email listing their items instead of one email per item.
Cron schedule (UTC)	<code>0 7 * * *</code>	When the sweep runs: five fields (minute hour day month weekday) in UTC. <code>0 7 * * *</code> is 07:00 UTC every day; <code>30 5 * * 1-5</code> is 05:30 UTC on weekdays. Invalid expressions are refused.
Review lead days	7	Reviews (risk, policy, data privacy), control tests and awareness programs due within this many days are reminded as "due". Earlier ones are "overdue". 0 to 365.
Exception lead days	14	Approved risk, compliance and policy exceptions expiring within this many days are flagged. 0 to 365.
Organization time zone	<code>America/New_York</code>	The IANA zone that decides which calendar day it is for everyone: due and overdue (My Tasks, control status, lists, reports, the AI assistant), expiry of exceptions and evidence, new review and test dates, POA&M opened and due dates, the sweep's due and overdue buckets, form defaults and export file names. Each person's own zone still decides how times are shown. Stored as <code>organization.timeZone</code> ; audited with the scheduler settings. Choose the zone your staff and auditors work in; the cron schedule stays in UTC.

Click **Save settings** to store them. Saving applies a changed **Cron schedule (UTC)** to the job queue at once, and the confirmation reads "Scheduler settings saved. The new schedule is active." If the queue cannot be reached (for example because the worker has never run), the save still succeeds and a warning says "The new schedule takes effect when the background worker next starts"; start or restart the worker in that case:

```
docker compose restart worker
```

The digest switch and the lead days apply at the next run.

Run sweeps now runs the whole sweep at once and reports how many items it found, for how many recipients, and how many digests it queued. It sends real email. An item already reminded in the last 20 hours is skipped, so running it next to the scheduled run sends nothing twice. A reminder goes only to a person who can still read the record it names. The run is recorded in the Audit Log.

7.7 Notification rules

Fixed notifications go to record owners on fixed events. Notification rules let an administrator email other people about other events. The **Rules** tab and every rule action are for administrators only. A new or upgraded installation has no rules.

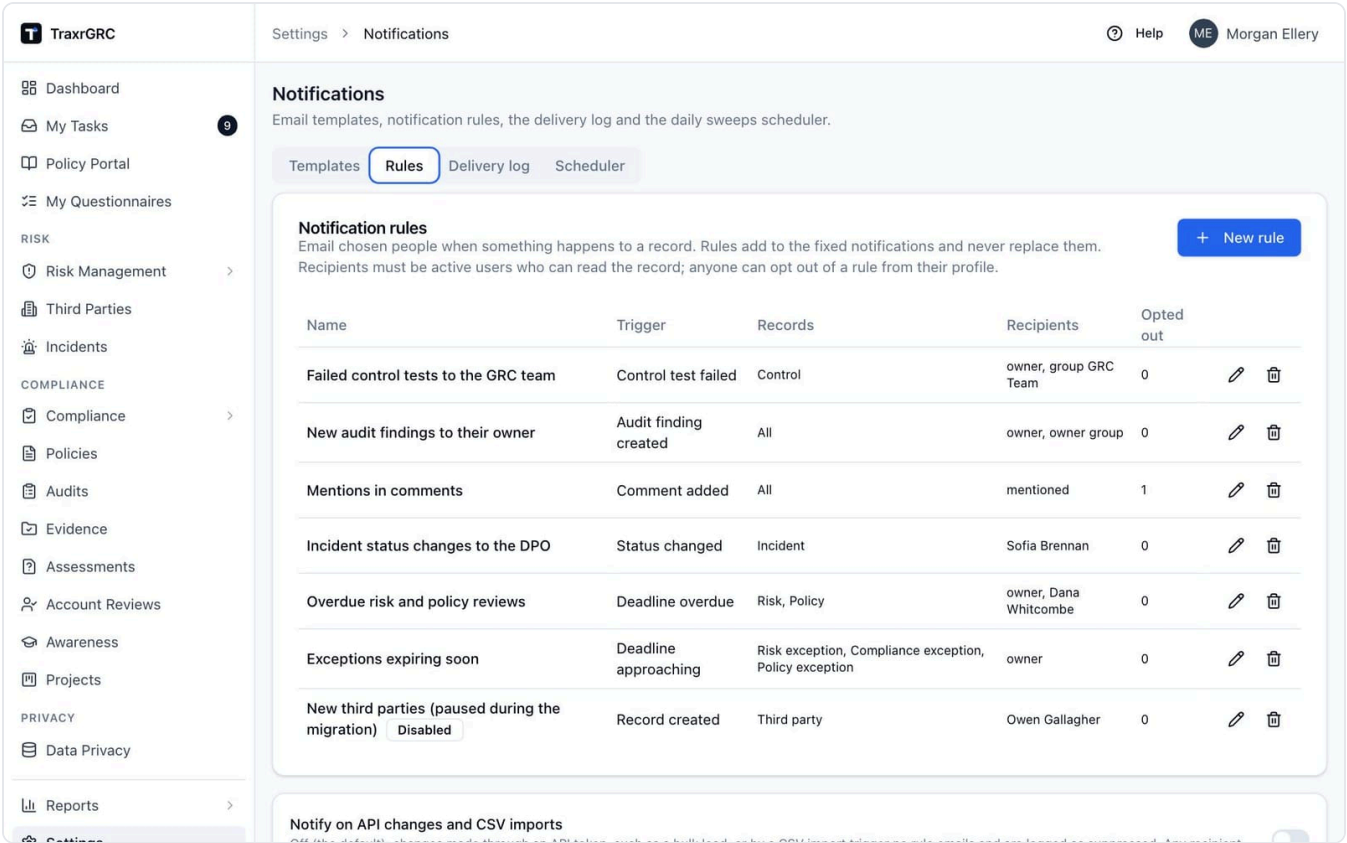


Figure 7.5. The Rules tab listing the demo's notification rules with their triggers, records, recipients and opt-out counts.

7.7.1 Creating a rule

1. Click **New rule**. The **New notification rule** dialog opens.
2. Fill in the fields and click **Create rule** (**Save rule** when editing).

New notification rule

×

Rule emails name the record, the event and a link, never the record's content. Each recipient gets one only if they may read the record when it is sent.

Name

Trigger

Records

None ticked means every record type this trigger supports.

☐ Risk
☐ Control
☐ Compliance requirement
☐ Compliance exception
☐ Policy

Only when the record's status is

None ticked means any status. For a status change, the new status.

☐ Open
☐ Closed
☐ Archived
☐ Planned
☐ Active

Recipients

☒ The record's owner
☐ Members of the record's owner group
☐ People @mentioned in the comment

Figure 7.6. The New notification rule dialog with the trigger, record types, status condition and recipient choices.

Field	What it means
Name	Up to 120 characters. Shown in the email, in the delivery log and on each person's My profile page.
Trigger	Comment added, Record created, Status changed, Audit finding created, Control test failed, Deadline approaching or Deadline overdue.
Records	Risk, Control, Compliance requirement, Policy, Risk exception, Compliance exception, Policy exception, Incident, Project, Third party, Audit finding, as far as the trigger applies to them. None ticked means every kind the trigger supports.
Only when the record's status is	Optional. For Status changed , the new status; for other triggers, the status at the time. None ticked means any status.
Recipients	Any mix of: the record's owner; the members of the record's owner group; people @mentioned in the comment (Comment added only); named users; members of chosen groups; members of every group carrying a chosen role. Active users only; no outside addresses. At least one recipient is required.
Enabled	Untick to pause the rule.

7.7.2 How rules are delivered

- **Event triggers** (comments, creation, status changes, findings, failed tests) are read from the Audit Log every minute, so an email follows the change within a minute or two.
- **Deadline triggers** run with the daily sweep, using the scheduler's **Review lead days** and **Exception lead days**. A deadline rule emails once when a deadline enters the lead window and once when it passes; moving the date starts again.
- **Access is checked twice**: when the rule fires and again just before sending. A recipient who may not open the record gets nothing (`no-access`).
- **Nobody is emailed about their own change**.
- **Each person gets at most 20 rule emails an hour**; the rest are suppressed (`rate-limit`).
- **Content is minimal**: the record's title or reference, what happened and a link. Comment text, descriptions and other fields are never included. An incident recorded as a personal data breach is named only by a short reference, since its title can name affected people.
- **Retries**: a rule email that the mail server refuses is tried four times, a minute or more apart, before it is marked **Failed**. One left in **Sending** by a crash is marked failed after 15 minutes and never resent.
- **Opt-outs**: anyone can switch a rule off for themselves on **My profile**. The **Opted out** column counts them. An administrator cannot override an opt-out.

7.7.3 Bulk changes and rules

Changes made through an API token (including the bundle loader) and through a CSV import send no rule emails by default; the delivery log lists them as **Suppressed** with the reason `api-write`. This keeps a data load from flooding people's inboxes. To include them, turn on **Notify on API changes and CSV imports** below the rules list. The hourly cap still applies. Fixed notifications (reminders, decisions) are not affected by this switch.

7.8 Scheduled reports delivery

Scheduled reports are set up by users with Reports rights on **Reports → Scheduled** (see the User Manual, Report Center). For the administrator:

- Each scheduled report email uses the `report.scheduled` template and is sent through the same SMTP settings and worker. The worker checks for due schedules every minute.
- Each recipient's copy is prepared with that recipient's permissions at sending time, so two people can get different files.
- Files over 10 MB are not attached; the email links to the report instead.
- A schedule missed while the worker was stopped runs once when it comes back.
- Recipients are active users only, and each can unsubscribe on **My profile**.
- Failed sends are retried like rule emails.
- Administrators see every scheduled report email in the delivery log, with the schedule's name in **Rule or schedule** and the attachment name.

A new or upgraded installation has no schedules. On an upgraded installation, add Create, Update and Delete on Reports to the roles that should manage schedules (see Users, Groups and Roles).

7.9 The delivery log

The **Delivery log** tab lists every email, newest first, with **Created, Template, To, Rule or schedule, Subject, Status, Error** and **Sent at**. Filter by status and search across the recipient, subject, template key and error. Users who are not administrators do not see rule or scheduled report emails.

Status	Meaning
Queued	Waiting for the worker.
Sending	A rule or report email being sent.
Sent	Accepted by the mail server.
Failed	The mail server refused it or could not be reached; Error holds the reason.
Skipped	A fixed notification deliberately not sent: <code>template-disabled</code> , <code>template-not-found</code> , <code>no-recipient</code> , <code>recipient-unavailable</code> (the user is inactive or gone), or "sent in digest" when the item went into a digest.
Suppressed	A rule or scheduled report email deliberately not sent: <code>no-access</code> , <code>opted-out</code> , <code>api-write</code> , <code>rate-limit</code> , <code>recipient-inactive</code> , <code>record-deleted</code> , <code>rule-disabled</code> , <code>template-disabled</code> , <code>schedule-paused</code> , <code>schedule-deleted</code> or <code>report-unavailable</code> .

A fixed notification that fails is not retried: it stays **Failed**. Review, exception, control test and awareness reminders come again with the next daily sweep while the item is still due. Reminders sent once per state (data processing agreements, transfer impact assessments and data mapping questionnaires) and event emails such as `risk.appetite.breach` may not be sent again, so fix the mail server and tell the people concerned.

7.9.1 Troubleshooting

Symptom	Likely cause and fix
Every row Failed , error mentions <code>localhost</code> or <code>ECONNREFUSED</code>	SMTP was never configured. Fill in Settings → Integrations → SMTP .
Rows stay Queued	The worker is not running. Check <code>docker compose ps</code> and <code>docker compose logs worker</code> .
Failed with an authentication error	Wrong username or password, or the password was cleared by a host or port change. Type it again and save.
Failed with a TLS or STARTTLS error	Port and Use TLS do not match the server: 465 needs Use TLS on, 587 needs it off. A relay without STARTTLS cannot take a username.
Sent , but nothing arrives	The mail was accepted and then filtered or rejected downstream. See deliverability below and check the relay's own logs.
Reminders arrive at the wrong hour	The cron is in UTC. If saving showed the warning that the schedule takes effect when the worker next starts, restart the worker.
Links in emails point to the wrong address	<code>APP_URL</code> is wrong. Fix it in the deployment's <code>.env</code> and restart.
Nobody gets rule emails after a data load	Expected: API and import changes are suppressed (<code>api-write</code>).

7.10 Deliverability

TraxrGRC hands mail to your SMTP server; whether it reaches inboxes depends on how that server and your domain are set up.

- **Send through your organization's relay** (Microsoft 365, Google Workspace, or your own mail gateway), never directly to the internet from the server.
- **Use a From address on your own domain** that the relay may send as, for example a shared mailbox or a send-as permission for `grc@example.com`.
- **Make sure SPF, DKIM and DMARC cover the relay** for that domain. A From address on a domain whose SPF does not list the relay is often rejected or filed as junk, and the delivery log still shows **Sent**.
- **Allow the sender** in your mail filtering if reminders land in junk, and tell staff to expect mail from it.

- **Assessment emails go to outside contacts.** Their mail systems apply their own filtering; a From address on your main domain with DMARC alignment gives the best chance.
- **Volume is modest** (sweeps once a day, rules capped at 20 per person an hour), but a first sweep after loading a large register can queue many reminders at once. Check the relay's sending limits before go-live.

Loading and Importing Data

This chapter covers bringing data into TraxrGRC: what a new installation already holds, CSV imports per module, compliance framework packages, the bundle loader for a full migration over the API, a step-by-step plan for a first-time migration, and the demonstration data that must never be loaded on a production instance.

8.1 Choosing a method

Method	Best for	Runs as	Writes
CSV import (Import button on a list)	Registers kept in spreadsheets; bulk updates	The signed-in user, in the browser	Risks, controls, assets, policies, third parties, incidents, projects, audit findings; new risk, compliance and policy exceptions
Account review CSV (Import CSV on a review program)	The accounts to review	The signed-in user	Review items
Compliance package import (Compliance → Packages → Import)	A framework's requirement tree	The signed-in user	One package, its items and an analysis per item
Bundle loader (<code>scripts/load-bundle.ts</code>)	A full, repeatable migration with requirement mappings	An API token	Risk scale, classifications, a package, controls, policies, risks, requirement mappings and statuses
REST API	Anything else, and ongoing integrations	An API token	Most records (see API Access and Integrations)

Every method goes through the same services and checks as the screens: the same validation, the same permission rules (including *own*-level access), and an Audit Log entry and History entry for every record.

8.2 What a new installation already contains

The `migrate` service adds reference data on every start, without changing or deleting anything that exists:

- the four system roles (GRC Manager, Risk Owner, Auditor, Data Liaison);
- four risk classifications, the 5 × 5 likelihood and impact scales, the matrix and four appetite bands (only when those tables are empty);
- every email template (missing keys only);
- the shared dashboard layout, kept on the shipped default until someone changes it (see Bootstrapping the first administrator);
- the default data mapping questionnaire, "Processing activity" (only when there are no questionnaire templates).

Nothing else: no users besides the first administrator, no groups, packages, labels, custom fields, incident classifications, business units or records.

8.3 CSV imports

Most module lists have an **Import** button beside **New**. On Risks, Policies and Compliance a menu offers the record type; audit findings are imported from the Audits page. The button appears only to people who may create or update those records (for exceptions, update on the parent records). The User Manual describes the dialog step by step (Getting Started, Importing from CSV); this section is the administrator's reference.

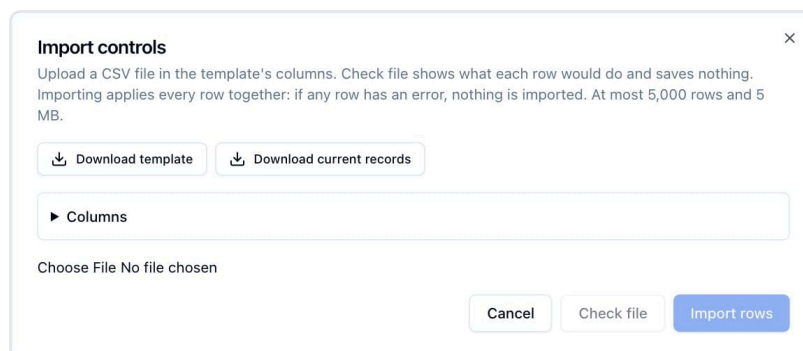


Figure 8.1. The Import controls dialog with the Download template and Download current records buttons.

8.3.1 Record types and columns

Import	Page	Permission	Columns	Custom fields
Risks	Risks	<code>risks:create</code> or <code>risks:update</code>	<code>id</code> , <code>title</code> *, <code>description</code> , <code>classification</code> *, <code>inherent_likelihood</code> *, <code>inherent_impact</code> *, <code>asset</code> , <code>business_unit</code> , <code>owner</code> , <code>owner_group</code> , <code>review_frequency_months</code> , <code>status</code>	Yes
Controls	Controls	<code>controls:create</code> or <code>controls:update</code>	<code>id</code> , <code>name</code> *, <code>description</code> , <code>type</code> , <code>status</code> , <code>owner</code> , <code>owner_group</code>	Yes
Assets	Assets	<code>assets:create</code> or <code>assets:update</code>	<code>id</code> , <code>name</code> *, <code>description</code> , <code>business_unit</code> , <code>owner</code> , <code>owner_group</code>	No
Policies	Policies	<code>policies:create</code> or <code>policies:update</code>	<code>id</code> , <code>title</code> *, <code>description</code> , <code>owner</code> , <code>owner_group</code> , <code>review_frequency_months</code>	Yes
Third parties	Third Parties	<code>thirdParties:create</code> or <code>thirdParties:update</code>	<code>id</code> , <code>name</code> *, <code>description</code> , <code>type</code> , <code>status</code> , <code>website</code> , <code>owner</code> , <code>owner_group</code>	Yes
Incidents	Incidents	<code>incidents:create</code> or <code>incidents:update</code>	<code>id</code> , <code>title</code> *, <code>description</code> , <code>classification</code> , <code>opened_at</code> , <code>asset</code> , <code>third_party</code> , <code>owner</code> , <code>owner_group</code> , <code>status</code> , <code>resolution</code>	Yes
Projects	Projects	<code>projects:create</code> or <code>projects:update</code>	<code>id</code> , <code>title</code> *, <code>description</code> , <code>status</code> , <code>deadline</code> , <code>owner</code> , <code>owner_group</code>	Yes
Audit findings	Audits	<code>audits:create</code> or <code>audits:update</code>	<code>id</code> , <code>audit_plan</code> *, <code>title</code> *, <code>description</code> , <code>status</code> , <code>deadline</code> , <code>control</code> , <code>owner</code> , <code>owner_group</code>	No

Import	Page	Permission	Columns	Custom fields
Risk exceptions	Risks	risks:update	risk *, title *, description *, owner, approver, expires_at *	No
Compliance exceptions	Compliance	compliance:update	package *, requirement *, title *, description *, owner, approver, expires_at *	No
Policy exceptions	Policies	policies:update	policy *, title *, description *, owner, approver, expires_at *	No

* Required when a row creates a record.

Allowed values:

Column	Values
Risk <code>status</code>	<code>open</code> , <code>closed</code> or <code>archived</code>
Risk <code>inherent_likelihood</code> , <code>inherent_impact</code>	The level's value or name from Risk Settings
Control <code>type</code>	<code>policy</code> , <code>process</code> , <code>technical</code> , <code>physical</code> or <code>other</code> (<code>process</code> when blank)
Control <code>status</code>	<code>planned</code> , <code>active</code> or <code>retired</code> (<code>active</code> when blank)
Third party <code>type</code>	<code>vendor</code> , <code>partner</code> , <code>customer</code> or <code>other</code> (<code>vendor</code> when blank)
Third party <code>status</code>	<code>active</code> , <code>offboarding</code> or <code>terminated</code>
Incident <code>status</code>	<code>open</code> or <code>closed</code> ; closing needs a <code>resolution</code>
Project <code>status</code>	<code>planned</code> , <code>in_progress</code> or <code>completed</code> (<code>planned</code> when blank)
Audit finding <code>status</code>	<code>open</code> , <code>in_remediation</code> or <code>closed</code> (updates only)
<code>review_frequency_months</code>	1 to 120 (12 when blank on a new record)
Dates (<code>opened_at</code> , <code>deadline</code> , <code>expires_at</code>)	YYYY-MM-DD

Values like statuses are matched without regard to case, spaces or underscores.

8.3.2 File rules

- **Header row.** The first row names the columns. Matching ignores case, spaces and punctuation, so `Owner Group` matches `owner_group` . Columns the import does not know are listed and ignored; a column named twice refuses the file.
- **Encoding.** Save the file as UTF-8 ("CSV UTF-8" in Excel). Other encodings garble accented characters.
- **Create or update.** A row with an `id` updates that record; a blank `id` creates one. Third parties without an id are matched by name, which is unique. Exceptions can only be created. Nothing is ever deleted.
- **Blank cells keep values.** A blank cell leaves the existing value unchanged, so a file can carry only the columns you want to change. A value cannot be cleared by import.
- **People and lookups.** `owner` and `approver` are active users, by email address or full name (use the email when two people share a name). `owner_group` , `classification` and `business_unit` are named as they appear in the app. Linked records (`asset` , `third_party` ,

`control`, `risk`, `policy`, `audit_plan`) take a name or an id and must be records the importer can read; a name that matches several records asks for the id.

- **Custom fields** are columns headed `Custom: <field name>`. Checkbox fields take yes or no; multi-select fields separate options with a semicolon.
- **Limits.** At most 5,000 data rows and 5 MB per file. Split larger files.
- **Formulas are stored as text.** A cell beginning with `=`, `+`, `-`, `@`, a tab or a carriage return is stored as plain text and never evaluated. Every CSV export prefixes such values with a single quote so a spreadsheet does not run them, and the import removes that quote again, so an exported file re-imports unchanged.

8.3.3 What an import does

1. **Check file** runs every row through the record's own service inside a database transaction and then rolls it back. The result table shows what each row would do (**Create**, **Update** with the changed columns, **Unchanged**, or **Error** with every reason by line number). Nothing is saved.
2. **Import rows** runs the same again and keeps the result only when every row succeeds. If any row fails at that moment, nothing is imported.
3. Each record's History shows the change, marked as coming from "CSV import of" the file name, and one summary Audit Log entry (action `import`) records the file, the row count and the outcome counts.
4. Notification rules ignore imported changes unless **Notify on API changes and CSV imports** is on (see Notifications and Email). Fixed reminders are unaffected: once imported, records are reminded like any other.

Download current records gives the records the importer can read, in the template's columns with ids filled in (at most 5,000). Edit it and import it back for bulk updates.

What CSV imports do not cover: policy content and approvals, risk treatments and reviews, control testing schedules, personal data breach details on incidents, the data map fields on assets, business units, audit plans, requirement mappings and anything in the privacy module. Use the app, the bundle loader or the REST API for those.

8.4 Compliance framework packages

A package is one framework (NIST SP 800-53, ISO/IEC 27001, the GDPR, a contract's security schedule) as a tree of headings and requirements. Each requirement gets an analysis, where its status is assessed.

8.4.1 Importing a package

Compliance → Packages → Import (needs `compliance:create` at all) reads a CSV or JSON file, previews the tree, and creates the package with **Commit import**. The file is refused as a whole when

any code or name is empty, a code repeats, a `parent_code` names a code not in the file, or parents form a cycle.



Figure 8.2. The Import compliance package dialog, before a file is chosen.

Format	Content
CSV	Header with <code>code</code> and <code>name</code> , optionally <code>description</code> , <code>parent_code</code> (also <code>parent</code>) and <code>kind</code> . One row per item. The package name, standard and version are typed in the dialog.
JSON	<code>{ "name", "standard", "version", "description", "items": [{ "code", "name", "description", "parentCode", "kind" }] }</code> . Name and at least one item are required.

`kind` is `requirement` (blank means requirement; counts toward compliance), `additional` (listed, counted separately) or `heading` (structure only). Limits: up to 10,000 items, codes up to 100 characters, names up to 300.

Every package exports from its row on the Packages page as JSON in exactly this shape, so a package can be copied between instances.

The import dialog sends the whole package to the server in one request. Screen requests are limited to 1 MB by the web framework, so a very large framework file may be refused there. Load such a file through the REST API instead, which accepts the same JSON:

```
curl -X POST https://grc.example.com/api/v1/compliance/packages \
-H "Authorization: Bearer $GRC_API_TOKEN" \
-H "Content-Type: application/json" \
--data-binary @package.json
```

The token needs `compliance:create` (see API Access and Integrations).

8.4.2 Framework content shipped with the source

TraxrGRC does not include licensed frameworks. Two public packages ship in the source and the container image, in the import JSON format, and can be imported on any instance:

File	Content
<code>prisma/demo/nist-800-53-moderate.json</code>	NIST SP 800-53 Rev 5 Moderate baseline: 18 families as headings, 287 controls and enhancements with their control statements. Public domain in the United States. Provenance in <code>prisma/demo/README.md</code> .
<code>prisma/demo/gdpr.json</code>	The GDPR: 11 chapters as headings, 99 articles, 40 of them requirements and 59 additional. Descriptions are plain-language summaries written for the package, not the legal text.

To copy one out of the running container and import it:

```
docker compose cp web:/app/prisma/demo/nist-800-53-moderate.json .
```

Then import the file with **Compliance → Packages → Import**. These files only create a package; they do not load any demo records.

For other frameworks, use content you are licensed to use: a purchased toolkit, the standard body's machine-readable release, or an export from your previous tool, converted to the CSV or JSON shape above.

8.4.3 Mapping requirements

A requirement's analysis links it to the controls, policies and risks that address it, and to remediation projects. Mappings can be made:

- in the app, on the requirement's page (see the User Manual, Compliance Management);
- in bulk with the bundle loader's `requirements` section (below);
- through the REST API, on `/api/v1/compliance/analyses`.

CSV imports do not map requirements.

8.5 The bundle loader

`scripts/load-bundle.ts` loads a complete set of existing GRC data into an instance over the REST API: a risk scale, classifications, a compliance package, controls, policies, risks, and each requirement's mappings and status. It is built for go-live and for running again whenever the source changes. The loader is generic; a separate exporter, kept with the client's data and never in the TraxrGRC repository, turns the client's registers into a bundle file. The full format reference is `docs/LOADING.md`.

8.5.1 Before you run it

1. **A workstation with the source.** The loader is not in the container image. Use a checkout of the TraxrGRC source at the same version as the instance, with Node.js 22 and its dependencies installed (`npm ci`). The workstation only needs HTTPS access to the instance.
2. **Outside approvals, if the bundle has them.** Policies with `effectiveDate` and `signatories` need **Settings → Policies → Allow recording outside approvals** turned on.
3. **An API token** created by an administrator (a token never exceeds its creator's rights), with these scopes:

Module	Rights	Why
Risks	Read, Create, Update	Risks, treatments, reading the risk scale
Controls	Read, Create, Update	Controls; reading them for mappings and treatments
Policies	Read, Create, Update, and Review when the bundle records outside approvals	Policies and approvals
Compliance	Read, Create, Update	Package, analyses and mappings
Settings	Create, Update	Classifications and the risk scale

Trimming this list breaks the load: scopes bound everything a request does, so mapping fails without Read on Controls, Policies and Risks. Give the token a short expiry and revoke it after go-live.

8.5.2 Running it

```
export GRC_API_TOKEN=aat_...  
npm run load -- bundle.json --url https://grc.example.com          # dry run  
npm run load -- bundle.json --url https://grc.example.com --apply  # write
```

Option	Meaning
<code>--url</code>	The instance's address. Required.
<code>--apply</code>	Write. Without it the loader only reads the instance and reports what it would do.
<code>--token-file</code> <code><path></code>	Read the token from a file, keeping it out of the shell history. Otherwise <code>GRC_API_TOKEN</code> is used.
<code>--ca-file</code> <code><path></code>	Trust a private certificate authority, for an instance behind an internal CA (for example Caddy with <code>tls internal</code>).

The bundle is validated first; any problem stops the run before anything is read or written, listing up to 50 problems. The dry run then prints, per record kind, how many records it would create, update or leave unchanged, plus warnings. `--apply` does the same and writes. The command exits with status 1 if any write failed, so it can run in a pipeline.

Every write goes through the API, so it is validated, recorded in the Audit Log against the token, and sends no rule emails unless **Notify on API changes and CSV imports** is on.

8.5.3 Running it again

Each record in a bundle carries a `ref`, the source's permanent identifier, and its title or name starts with that ref followed by a space (`R-01 Ransomware reaches...`). A later run finds records by that prefix and updates them. Nothing is ever deleted, and owners are never set (assign them in the app). Highlights of the re-run rules:

- The risk scale is replaced only when it differs, and never while a risk uses a level the new scale drops.
- An existing package is left alone; item codes missing from it are reported.
- Controls get their name, description, type and status updated.
- A draft policy is updated; a published policy gets new content only when the bundle carries a later effective date (a reissue).
- Risks get title, description, classification, status and inherent levels updated; a residual that no longer matches is reported, for adjustment through the risk's treatments in the app.
- Requirement mappings are only added, so mappings made by hand survive.
- A requirement status is set only while its analysis is still Not reviewed; an assessment made in the app always wins.

The full table is in `docs/LOADING.md`.

8.5.4 Bundle format in brief

```
{
  "format": "grc-bundle",
  "version": 1,
  "source": "Exported from the 2026 risk register and SSP",
  "riskScale": { "likelihood": [], "impact": [], "cells": [], "bands": [] },
  "package": { "name": "...", "standard": "...", "version": "...", "items": [] },
  "classifications": [{ "name": "Availability" }],
  "controls": [{ "ref": "AC-2", "name": "AC-2 Account Management" }],
  "policies": [{ "ref": "POL-110", "title": "POL-110 Acceptable Use" }],
  "risks": [{ "ref": "R-01", "title": "R-01 Ransomware halts production" }],
  "requirements": [{ "code": "AC-2", "controls": ["AC-2"], "policies": ["POL-110"], "risks":
}
```

Rules the validator enforces: refs unique per record type and up to 60 characters; titles and names start with their ref; free text up to 100,000 characters; residual likelihood and impact never above inherent; every risk's classification listed in `classifications`; with a `riskScale`, every risk's levels on that scale and a matrix cell for every pair; requirements only with a `package`, naming its item codes and refs in the same bundle; policies published from the bundle carry their content.

Each risk needs a `residualBasis`, which becomes an implemented mitigate treatment carrying the residual rating, and may carry a `plannedTreatment`.

8.6 First-time migration

Whether the data comes from spreadsheets or another GRC tool, load it in an order where everything a record refers to already exists. There is no direct importer for another vendor's database: export that tool's registers to CSV (or write an exporter to a bundle) and load them with the methods above.

1. **Freeze the target.** Configure branding, SMTP and sign-in, but tell users the instance is not live yet. Decide whether rule emails stay suppressed during the load (the default).
2. **Settle the risk method.** Set classifications, magnitudes, matrix and appetite bands in Risk Settings, or put them in the bundle's `riskScale`. Map the old tool's ratings to the new values in writing.
3. **People.** Create users and groups, or let SSO, LDAP or SCIM create them (see Authentication and Single Sign-On). Owners in every import must be active accounts, and owner groups must exist.
4. **Supporting lists.** Create business units (Risk Management → Business Units, or `/api/v1/business-units`), incident classifications, labels, custom fields and audit plans. None of these has a CSV import.
5. **Catalogs.** Import controls, assets and third parties.
6. **Frameworks.** Import compliance packages.

7. **Policies.** Import titles, owners and review frequencies by CSV, then add content and approvals in the app, or load them by bundle with content and outside approvals.
8. **Registers.** Import risks (linking assets and business units), incidents (linking assets and third parties), projects, and audit findings (naming their plan and control).
9. **Exceptions.** Import risk, compliance and policy exceptions once their parent records exist. They arrive as requested; decide them in the app.
10. **Mappings and treatments.** Map requirements and add risk treatments in the app, by bundle, or over the API.
11. **Check.** Run the Report Center reports, compare counts with the source, and open a sample of records. Review the Audit Log entries for the load.
12. **Before go-live,** look ahead at reminders: the first daily sweep after the load emails owners about every review, control test and expiring exception that is due or overdue in the loaded data. Correct wrong dates first, and tell owners to expect the emails.
13. **Revoke migration tokens** and switch on any notification rules.

Keep the source files and the loader output with the migration records; the Audit Log entries carry the token and file names that tie each change back to them.

8.7 Demo data

The source includes a sales demonstration: a fictional company, Quillmont Systems, with records in every module, NIST SP 800-53 Moderate and the GDPR as packages, notification rules, scheduled reports and eight sign-in personas. It is loaded by `prisma/demo-seed.ts` (`npm run db:demo`) and described in `docs/DEMO.md`.

Never load the demo on an instance that will hold real data. It creates fictional people with working sign-in accounts (one of them an administrator), dozens of fictional records across every module, notification rules and scheduled reports that email those people, and sets branding and settings (product name "TraxrGRC", outside approvals on). Its records are created through the normal services, so they are written into the hash-chained Audit Log, which cannot be edited or pruned for a year. There is no way to remove the demo afterward; the only clean path is a new, empty database.

Two safeguards exist:

- **The demo seed refuses an instance in use** (`prisma/demo/guard.ts`). Before writing anything, it counts rows in the tables an organization fills (risks, controls, assets, business units, packages, policies, third parties, incidents, incident classifications, audit plans, projects, awareness programs, records of processing, account review programs, assessment templates, custom fields, labels, groups, notification rules, report schedules and API tokens) and refuses if any has rows, if any user other than an administrator exists, or if a demo account already exists. It never deletes. A freshly installed production instance is still empty, though, and the guard accepts it, so the real protection is not running the command there.
- **The older development seed refuses production.** `prisma/seed.ts` (`npm run db:seed`, also what `prisma db seed` runs) begins by deleting the data in every table. It refuses to run when

`NODE_ENV` is `production`, as it is in the container image, unless `ALLOW_DEMO_SEED=1` is set. Never set that variable on a real instance.

To show the product, run the demo on a separate instance with its own database and its own address (see `docs/DEMO.md`), and keep its accounts file private.

CHAPTER 9

API Access and Integrations

This chapter covers everything that connects TraxrGRC to other systems: API tokens, the REST API, the Model Context Protocol server for AI assistants, and the practices that keep machine access safe.

9.1 Integration points

Integration	Address	Authenticates with	Turned on by
REST API	<code>/api/v1/...</code>	API token	Always available
OpenAPI document	<code>/api/v1/openapi</code>	None (public)	Always available
AI assistants (MCP)	<code>/api/mcp</code>	API token, or Microsoft Entra ID access token	Settings → AI assistants
SCIM provisioning	<code>/scim/v2</code>	SCIM token	Settings → Provisioning (see Authentication and Single Sign-On)
LDAP and SAML sign-in		Directory or identity provider	Settings → Integrations (see Authentication and Single Sign-On)
Outbound email	Your SMTP server	SMTP credentials	Settings → Integrations (see Notifications and Email)
Outside assessments	<code>/assess/<token></code>	A per-request link	Sending an assessment (see the User Manual, Online Assessments)
Health check	<code>/api/health</code>	None (public)	Always available; answers <code>{"status":"ok"}</code> or 503

TraxrGRC has no outbound webhooks and sends nothing to outside addresses except assessment emails, by design. Systems that need to react to changes read the REST API on a schedule.

9.2 API tokens

Settings → API Tokens (address `/settings/api-tokens`) issues bearer tokens for the REST API and the MCP server.

Action	Needs
Open the page	<code>apiTokens:read</code> . With it at <i>all</i> you see every token; otherwise only your own.
Create a token	<code>apiTokens:create</code>
Revoke your own token	<code>apiTokens:update</code>
Revoke someone else's	<code>apiTokens:update</code> at <i>all</i>

None of the built-in roles holds API token rights, so by default only administrators can issue tokens.

The list shows each token's **Name**, **Scopes**, **Created by**, **Created**, **Last used**, **Expires** and **Status** (**Active**, **Expired** or **Revoked**). **Last used** is updated at most once a minute.

9.2.1 Creating a token

1. Click **Create token**. The **Create API token** dialog opens.
2. Enter a **Name** that says what uses it and who owns it, for example "SIEM export (Security Ops)". Up to 120 characters.
3. Optionally set an **Expiry date**. The label states the longest lifetime allowed (365 days unless changed on the Sessions page). Blank gets the longest lifetime; a later date is refused. When the maximum is 0, a blank date means the token never expires.
4. Tick the scopes: per module, **Read**, **Create**, **Update**, **Delete**, and **Review** where the module has it (Risks, Compliance, Policies, Data privacy). At least one is required.
5. Click **Create token**.
6. The **Token created** dialog shows the token once. Copy it into your secret store, then click **Done**. Only a SHA-256 hash is kept; a lost token cannot be recovered, only replaced.

Create API token

Grant read or write rights per module. The token is shown once after creation.

Name

CI compliance exporter

Expiry date (at most 365 days; blank uses the maximum)

mm/dd/yyyy

Module	Read	Create	Update	Delete	Review	Approve
Risks	☐	☐	☐	☐	☐	—
Compliance	☐	☐	☐	☐	☐	—
Policies	☐	☐	☐	☐	☐	—
Third parties	☐	☐	☐	☐	—	—
Incidents	☐	☐	☐	☐	—	—
Audits	☐	☐	☐	☐	—	—
Awareness	☐	☐	☐	☐	—	—
Projects	☐	☐	☐	☐	—	—
Data privacy	☐	☐	☐	☐	☐	☐
Data Liaison questionnaires	☐	☐	☐	☐	—	—
Account reviews	☐	☐	☐	☐	—	—
Assessments	☐	☐	☐	☐	—	—
Evidence	☐	☐	☐	☐	—	—
Custom fields	☐	☐	☐	☐	—	—

Figure 9.1. The Create API token dialog with the name, expiry date and scope matrix.

Creating a token is recorded in the Audit Log and emails every administrator a security alert naming the token, its scopes and expiry.

9.2.2 What a token can do

A token acts as the user who created it, and every request must pass both checks:

- **The token's scopes.** They bound everything the request does, beyond the endpoint it calls: creating a risk from a DPIA also needs Create on Risks, linking a vendor needs Read on Third parties, and mappings to modules the token cannot read come back hidden.
- **The creator's current rights**, including *own*-level access, checked on every request. A token never exceeds its creator. If the creator loses a right, the token loses it too; if the creator is deactivated or deleted, the token stops working (401).

A token works on the REST API and the MCP server. It does not work at the SCIM address, and SCIM tokens do not work on the API.

9.2.3 Expiry, revocation and restores

- An expired token is refused at once and shows **Expired**.
- Click the trash on a token's row and confirm **Revoke token?** to stop it. Revocation is immediate, cannot be undone, and is audited.
- **Restoring a backup snapshot revokes every API token and every SCIM token** (see Backup and Restore). Plan to issue new tokens to every integration, and a new SCIM token to the identity

provider, after a restore.

- Changing the maximum lifetime on the Sessions page affects new tokens only.

9.3 The REST API for administrators

Developers find the endpoints, parameters and payloads in the OpenAPI document and the User Manual chapter REST API. What an administrator needs to know:

- **Base address:** `https://<your address>/api/v1`. Requests and responses are JSON; lists page with `page` and `pageSize` (at most 200).
- **Authentication:** `Authorization: Bearer aat_...` on every request. Missing, unknown, revoked or expired tokens get 401; missing scopes or creator rights get 403.
- **OpenAPI document:** `/api/v1/openapi`, public, cached for five minutes, titled with the product name. Load it into a viewer or client generator. Each operation states the scope it needs.
- **Coverage:** risks (with classifications and the risk scale), controls, assets, business units, compliance packages and analyses, policies, third parties, incidents, audits, projects, awareness, data privacy, account reviews, assessments and custom fields. Not covered: users, groups, roles, settings pages, notifications, the Report Center, the dashboard, the Audit Log and backups.
- **Settings through the API:** creating risk classifications needs `settings:create`; replacing the risk scale (`PUT /api/v1/risk-scale`) needs `settings:update`. Reading both needs `risks:read`.
- **Auditing:** every change is recorded in the Audit Log naming the token and its creator. Every refusal is recorded too (module `security`, action `permission.denied`, with the reason: missing token, invalid, revoked, expired, scope, creator unavailable, creator rights or rate-limited), at most ten a minute per person or address.
- **Rate limit:** each token may make 600 requests a minute on `/api/v1` by default. Past that, requests get `429` with a `Retry-After` header giving the seconds until the token's one-minute window ends, and the body `{"error":"rate_limited"}`. Only the first refusal in each window is recorded (`permission.denied`, reason `rate-limited`). Set `API_RATE_LIMIT_PER_MINUTE` in `.env` to change the budget, or `0` to turn it off (see Configuration Reference). The count is kept in each web process, so it resets when `web` restarts. Large writes are also bounded by the record validation limits (for example 100,000 characters of free text).
- **Notifications:** API changes send no rule emails unless **Notify on API changes and CSV imports** is on.

A quick check that a token works:

```
curl -s -H "Authorization: Bearer $GRC_API_TOKEN" \
  "https://grc.example.com/api/v1/risks?pageSize=1"
```

9.4 AI assistants (MCP)

TraxrGRC includes a Model Context Protocol server at `/api/mcp`, so AI assistants (Microsoft Copilot Studio agents, GitHub Copilot in VS Code, Claude Code and other MCP clients) can search and read records and attach evidence on behalf of a user. The full reference, with setup for each client, is `docs/MCP.md`; this section covers the administrator's decisions.

TraxrGRC

Dashboard

My Tasks

Policy Portal

My Questionnaires

RISK

Risk Management

Third Parties

Incidents

COMPLIANCE

Compliance

Policies

Audits

Evidence

Assessments

Account Reviews

Awareness

Projects

PRIVACY

Data Privacy

Reports

Settings

< Collapse

Settings > AI assistants

Help Morgan Ellery

AI assistants (MCP)

Model Context Protocol endpoint

Lets AI assistants such as Microsoft Copilot Studio, GitHub Copilot in VS Code and Claude search and read records and upload evidence, always as a named user with that user's permissions. Every call is recorded in the audit log.

Allow AI assistants to connect

Turns on `https://grc.quillmont.example/api/mcp`. While off, the address answers 404. People connect with an API token they create under Settings, API tokens, or with Microsoft Entra ID sign-in below. Off by default. Only an administrator can change this.

Include privacy records

When off, processing activities, DPIAs, personal data breaches, transfer assessments and Data Liaison questionnaires are never returned or accepted through MCP, whatever the user's rights. Off by default.

Microsoft Entra ID sign-in

For shared Copilot Studio agents where each person must act as themselves. The assistant signs the person in with Entra ID and sends an access token; the token's sign-in name must match an existing active user here. No accounts are created.

Accept Entra ID access tokens

Needs an app registration in your tenant that exposes the scope below and issues v2 access tokens.

Directory (tenant) ID

00000000-0000-0000-0000-000000000000

The tenant GUID. Tokens from any other tenant are refused.

Application (client) ID

00000000-0000-0000-0000-000000000000

Of the app registration that exposes the API. v2 access tokens carry it as their audience.

Application ID URI

Leave empty to use `api://<client ID>`. Also accepted as a token audience.

Required scope

Mcp.Access

The delegated scope exposed by the app registration that every token must carry.

Server URL for the assistant: `https://grc.quillmont.example/api/mcp`
Protected resource metadata: `https://grc.quillmont.example/.well-known/oauth-protected-resource/api/mcp`
Copilot Studio reaches this server from Microsoft's cloud, so the instance must be on public HTTPS for it. A server on a private network works only with assistants that run on the same network, such as VS Code or Claude Code.

Save

Figure 9.2. The AI assistants (MCP) screen with the endpoint switches and the Microsoft Entra ID sign-in card.

9.4.1 Turning it on

Settings → AI assistants (address `/settings/mcp`) opens with `settings:read` ; only an administrator can save. Every save is audited and emailed to every administrator.

Setting	Default	Effect
Allow AI assistants to connect	Off	Turns on <code>/api/mcp</code> . While off, the address and its metadata documents answer 404.
Include privacy records	Off	While off, processing activities, DPIAs, personal data breaches, transfer impact assessments, DPA fields on vendors, system data fields on assets and Data Liaison questionnaires are never returned or accepted, whatever the user may see in the app.
Accept Entra ID access tokens	Off	Lets people sign in to a shared Copilot Studio agent as themselves. Needs the Directory (tenant) ID , Application (client) ID , optionally the Application ID URI (default <code>api://<client ID></code>) and the Required scope (default <code>Mcp.Access</code>).

There is no per-user switch: while the endpoint is on, anyone with an API token or an Entra sign-in can use it. Control access through who may hold tokens.

9.4.2 What an assistant can do

Nine tools, all acting as one named user with that user's live permissions:

Tool	Purpose
<code>search</code>	Text search across risks, controls, policies, requirements, vendors, assets, audit plans and findings, incidents, projects, business units and evidence (processing activities when privacy records are included)
<code>get_record</code>	One record in full, with key linked records and attachment details (including each file's SHA-256)
<code>list_risks</code> , <code>list_controls</code> , <code>list_compliance</code> , <code>list_policies</code>	Filtered lists
<code>my_open_items</code>	What is waiting on the user
<code>list_evidence</code>	Evidence and vendor reports with status, files, links and lock state
<code>create_evidence</code> , <code>link_evidence</code> , <code>lock_evidence</code>	Record evidence or a vendor's report, link it to the records it supports (or unlink it), and lock it; they need <code>evidence:create</code> or <code>evidence:update</code> , and unlocking stays in the app and the REST API
<code>upload_evidence</code>	Attach short evidence the assistant writes (up to 10 MB) to a record the user may update; for a control or compliance requirement it lands on new evidence linked to the record, which needs <code>evidence:create</code>
<code>request_upload_link</code>	A browser link, valid 15 minutes, for the user to upload up to 10 files (25 MB each) to a record

No tool deletes, unlocks, edits record fields or decides anything; the evidence tools only create evidence, change its links and lock it.

9.4.3 Authentication

- **API token:** the token's creator is the user, limited to the token's scopes. A read-only assistant needs Read scopes only; the two upload tools also need Update and Create on the target module, and the evidence tools Evidence Create or Update. In Copilot Studio an API key makes every person using the agent act as the token's owner, so use it only for personal agents.
- **Microsoft Entra ID:** each person signs in as themselves. The account must already exist and be active with the same email as the Entra sign-in name; MCP never creates or links accounts. An account with a password or administrator rights is matched only after it has signed in with SAML or an administrator has allowed linking on it (see Authentication and Single Sign-On).

9.4.4 Limits, auditing and reachability

- 120 tool calls a minute, 20 uploads an hour and 20 upload links an hour, per token (per user for Entra sign-ins), counted in each web process.
- Requests are POST only, JSON only, up to 16 MB, and a request whose `Origin` is not `APP_URL`'s is refused.
- Every tool call is recorded in the Audit Log (module `mcp`, action the tool name) with the user, the token or Entra sign-in, the address and the arguments without file content. Upload links are audited when issued and for each file.
- Copilot Studio calls from Microsoft's cloud and needs a public HTTPS address. A private instance works only with assistants on the same network (VS Code, Claude Code), and Node-based clients need `NODE_EXTRA_CA_CERTS` for a private certificate authority.

To connect Claude Code with a token, for example:

```
claude mcp add --transport http grpc https://grc.example.com/api/mcp \
  --header "Authorization: Bearer aat_..."
```

9.5 Service accounts and token practice

A token acts as its creator, so the account that creates it decides what the integration can ever do and whose name appears in the Audit Log.

Use a dedicated service account for each long-lived integration:

1. Create a user for the integration, for example `svc-siem@example.com`, with a recognizable name ("Service: SIEM export"). Leave **Administrator** off.
2. Create a role holding exactly the rights the integration needs, plus Read and Create on API tokens, and attach it to a group containing only the service account.
3. Sign in as the service account once, create the token with the same narrow scopes, and store it in your secret manager.
4. Take the API token rights off the role afterward if you want no one using that account to issue more. The existing token keeps working.

With **SSO only** turned on, password sign-in is refused for everyone except break-glass administrators, so a service account must then exist in your identity provider to sign in and create its token.

Tokens created by a person's own account stop working when that person leaves and their account is deactivated, which breaks the integration. Short-lived tokens for one-off jobs (such as the bundle loader) can come from an administrator's account, revoked when the job is done.

Good practice for every token:

- One token per integration, named for the integration and its owner.

- The fewest scopes that work; Read only where the integration only reads.
- An expiry date. Put the renewal in a calendar: create the new token, switch the integration over, then revoke the old one.
- Keep tokens out of command lines, scripts and repositories: use a secret manager, environment variables or `--token-file`.
- Review the API Tokens page regularly. Revoke tokens whose **Last used** is old or empty, and any whose owner has left.
- Watch the security alert emails for token creation you did not expect, and the Audit Log for `permission.denied` entries naming a token.
- After restoring a backup, issue new tokens: a restore revokes them all.

Audit Log and Evidence Integrity

TraxrGRC records every change to its records and configuration, every sign-in and refused request, in an append-only audit log protected by a hash chain. This chapter describes what is recorded, how to search and export it, how to verify its integrity, how retention works, and how to present it to an auditor.

10.1 What is logged

Every entry records:

Field	Content
When	The server time, to the millisecond (shown to the second)
Actor	The email of the person, <code>SCIM</code> , <code>system</code> , or the typed email of a failed sign-in
API token	The token id, when the change came through the REST API
IP	The client address, as reported through <code>TRUSTED_PROXY_HOPS</code>
Module	The module, for example <code>risks</code> , <code>users</code> , <code>auth</code> , <code>security</code>
Action	What happened, for example <code>create</code> , <code>update</code> , <code>delete</code> , <code>sign-in</code>
Entity and Entity ID	The record type and its identifier
Changes	A before and after record of the fields changed, where applicable, and the source when a change was made on someone's behalf (for example a CSV import or a questionnaire)

The categories recorded:

Category	Module and actions
Record changes	Each module's <code>create</code> , <code>update</code> and <code>delete</code> , plus workflow actions such as <code>publish</code> , <code>submit-for-approval</code> , <code>acknowledge</code> , <code>transition</code> , <code>close</code> , <code>reopen</code> , <code>review</code> , <code>attest</code> , <code>apply</code> and <code>import</code> ; for evidence, <code>link</code> , <code>unlink</code> , <code>lock</code> (an automatic lock has actor <code>system</code> or the person recording the test result, and <code>automatic: true</code>), <code>unlock</code> (with the reason) and <code>verify</code> (the file check's counts, naming any changed or missing file); attachment <code>create</code> entries carry the file's SHA-256
Users, groups and roles	<code>users</code> <code>create</code> , <code>update</code> , <code>delete</code> on users, groups and roles (role entries hold the full permission matrix; user updates name groups on both sides, while a create records group ids); <code>sessions.revoke</code> ; <code>sso.link</code> ; <code>timezone.update</code> ; <code>password.change</code> , <code>password.change.failed</code> ; <code>mfa.enroll.started</code> , <code>mfa.enabled</code> , <code>mfa.reset</code>
Sign-ins	<code>auth</code> <code>sign-in</code> (method, whether a break-glass administrator signed in with a password under SSO only, and <code>mfaPending</code> when the session waits for the two-step verification step), <code>sign-out</code> , <code>sign-in.failed</code> , <code>lockout</code> , <code>account.guarded</code> ; the two-step verification step as <code>mfa.verified</code> , <code>mfa.failed</code> , <code>mfa.locked</code> and, during setup, <code>mfa.enroll.failed</code>
Refused requests	<code>security</code> <code>permission.denied</code> : a screen action, API call, API token or SCIM request refused, with the module and right or the reason (including <code>rate-limited</code> for an API token over its request budget)
Settings	<code>settings</code> <code>update</code> for integrations (every field, secrets only as changed or unchanged, the IdP certificate as a SHA-256 fingerprint), sessions, separation of duties and other settings
Provisioning	SCIM user and group changes with actor <code>SCIM</code> and the token; <code>scimToken.create</code> , <code>scimToken.rotate</code> , <code>scimToken.revoke</code> ; <code>scim.link</code> , <code>scim.link-refused</code> , <code>scim.deprovision</code>
Audit log	<code>auditLog</code> <code>verify</code> (each integrity check and its result) and <code>auditLog</code> <code>delete</code> (each prune, with the cutoff and count)
Backups	Export, restore and backup failure entries (see Backup and Restore)

Failed sign-ins carry a reason:

Reason	Meaning
<code>unknown-user</code>	No account has that email
<code>no-password</code>	The account has no local password
<code>bad-password</code>	Wrong password
<code>inactive</code>	The account is deactivated
<code>locked</code>	Refused by the throttle; the detail names the lock (<code>pair-locked</code> , <code>ip-locked</code> , <code>account-guarded</code> , <code>passwordless-locked</code>)
<code>sso-only</code>	Password or LDAP sign-in refused because SSO only is on
<code>method-disabled</code>	LDAP or SAML is switched off
<code>ldap-rejected</code>	The directory refused the user or could not be reached; the detail says which
<code>ldap-no-mail</code>	The directory entry has no mail attribute
<code>link-refused</code>	LDAP or SSO asserted the email of an account that may not be linked
<code>saml-invalid</code>	The SAML response failed validation; the detail gives the check
<code>saml-replay</code>	The SAML assertion had already been used
<code>invalid-input</code>	Malformed request

No account is created for an unknown email by a failed sign-in.

To keep the log from flooding, failed sign-ins are recorded at most 60 a minute per address, and refused requests at most 10 a minute per person (or per address without a valid token). The next entry written records how many were dropped (`suppressedBefore`).

Changes are recorded in the same database transaction as the change itself wherever the change is transactional, so a change that commits always has its entry. A CSV import writes its entries together just before the import commits, each marked with the import as its source, plus one summary entry.

10.2 The Audit Log page

Open **Settings** → **Audit Log** (address `/settings/audit-log`) from its card on the Settings page; security alert emails link to it too. Viewing needs `auditLog:read` at All, held by administrators and by

the built-in GRC Manager and Auditor roles.

TraxrGRC Settings > Audit Log Help Morgan Ellery

Audit Log
Every mutating action in the application, with before/after diffs where applicable.

Integrity
Each entry carries a hash of its content and of the entry before it. Verifying recomputes the chain and shows the first entry that was changed, removed or reordered. The check is recorded in the audit log.
[Verify audit log](#)

Retention
Permanently delete audit entries older than the retention window, at least 365 days. Every administrator is told.
Older than (days): [Prune...](#)

Search... [Filters](#) [Export CSV](#) [Saved](#) [Columns](#)

When	Actor	IP	Module	Action	Entity	Entity ID	Change
2026-10-19 19:02:42 EDT	morgan.ellery@quillmont.example	203.0.113.24	auth	sign-in	user	cmvfv01ai000018tfuj3c...	View
2026-10-19 19:02:34	system	-	compliance	expire	complianceException	cmvfv0nfe02pwi8tfa6g...	View

Figure 10.1. The Audit Log with the Integrity and Retention cards above the table of entries.

Administrators see two cards above the table, **Integrity** and **Retention**; other readers see only the table.

10.2.1 Searching and filtering

- **Search...** matches the actor, entity type, entity ID and action.
- **Filters** opens a panel with a filter for each column: free text for **Actor**, **IP** and **Entity ID** (matching any part of the value), dropdowns for **Module**, **Action** and **Entity** whose options are the values present in the log, and a date picker for **When**, which shows the entries of that calendar day in your display time zone.
- The log opens sorted by **When**, newest first. Click a column heading to sort another way.
- **Saved** stores the current filters under a name, for example "Sign-in failures" (module `auth`, action `sign-in.failed`), for reuse.
- **Columns** hides and shows columns.

Useful filters:

To find	Filter
Every refused sign-in	Module <code>auth</code> , Action <code>sign-in.failed</code>
Lockouts	Module <code>auth</code> , Action <code>lockout</code> or <code>account.guarded</code>
Who changed a role	Module <code>users</code> , Entity <code>role</code>
Group membership changes	Module <code>users</code> , Entity <code>group</code>
Administrator rights granted	Module <code>users</code> , Entity <code>user</code> , then View diff for <code>isAdmin</code>
Refused requests	Module <code>security</code> , Action <code>permission.denied</code>
All changes to one record	Entity ID set to the record's id (from its address)
What one person did	Actor set to their email
Everything on one day	When set to that date
Activity from one address	IP set to the address, or its first part for a range
Two-step verification failures	Module <code>auth</code> , Action <code>mfa.failed</code>
Changes by provisioning	Actor <code>SCIM</code>
Integrity checks	Module <code>auditLog</code> , Action <code>verify</code>

10.2.2 Viewing a change

Click **View diff** in the **Changes** column to open **Change details**, with the **Before** and **After** values side by side, exactly as recorded.



Figure 10.2. The Change details dialog showing the Before and After values of an audit entry side by side.

10.2.3 Exporting

Export CSV downloads the entries matching the current search and filters, with the columns When, Actor, IP, Module, Action, Entity, Entity ID and Changes. Times are in UTC (ISO 8601). An export holds at

most 5,000 rows; when more match, a warning tells you so; narrow by day with the **When** filter, or by module, action, entity or actor, and export each part.

The **Changes** column of the export holds the recorded change as compact JSON, with its `before`, `after` and, where recorded, `source` values. It is empty for entries that record no change.

10.3 The hash chain

Every audit entry carries three values set by the database when the entry is inserted: a sequence number (its chain position), the hash of the previous entry, and its own hash. The hash is SHA-256 over the entry's content and the previous hash:

Hashed, in order	
Format tag <code>audit-v1</code>	
Previous entry's hash	All zeros for the first entry
Entry id, timestamp, actor email, module, action, entity type, entity ID, changes, IP, API token id	Each length-prefixed, so no value can shift into its neighbor

The actor's user id is deliberately outside the hash: deleting a user clears it, and a restore links it back, without breaking the chain.

Because each hash covers the one before it, changing, deleting or reordering any entry breaks every link after it. The database protects the chain in three layers:

1. **Insert trigger.** A database trigger computes the position and hashes under a lock, so concurrent writes form one linear chain and no application code can skip it.
2. **Append-only trigger.** Updates and deletes on the audit log are refused by the database, from the application or any other client, except the two maintenance operations below.
3. **Separate database role.** The application connects as a role (default `grc_app`) that may only read and insert audit entries, owns nothing, and cannot change or disable the triggers. Only the database owner can.

The two maintenance operations run as owner functions: pruning (below) and relinking actor ids after a restore.

Within these limits, the chain is tamper-evident to everyone except the database owner. Someone holding the owner credentials could, in principle, rewrite entries and recompute the chain after them. Keep the owner password (`DATABASE_URL` for the migrate container) out of daily use, and record the result of each verification outside TraxrGRC (below), so a rewritten chain disagrees with your record.

10.3.1 Verifying the chain

On screen, administrators:

1. Open `/settings/audit-log`.
2. In the **Integrity** card, click **Verify audit log**.
3. Read the result under the button:
 - "Intact: N entries verified (chain positions A to B)." Every entry from position A to B links correctly and matches its hash.
 - "Broken at entry <id> (<reason>) after N intact entries." The first entry that fails, or "at the end of the chain" when entries are missing from the end.

Reason	Meaning
<code>hash-mismatch</code>	The entry's content was changed after it was written
<code>prev-hash-mismatch</code>	The entry does not follow the one before it: an entry was replaced or reordered
<code>sequence-gap</code>	An entry was removed
<code>head-mismatch</code>	Entries were removed from the end of the chain
<code>unchained</code>	An entry exists without chain values
<code>before-anchor</code>	An entry older than the last prune remains, so the prune record was tampered with

Verification reads the whole log in a single consistent snapshot, in batches of 5,000, and is itself recorded as an `auditLog` `verify` entry with the result.

From the command line, on a Docker Compose deployment:

```
docker compose exec web ./node_modules/.bin/tsx prisma/audit-verify.ts
```

It prints the result as JSON, for example `{"ok":true,"rows":18234,"fromSeq":0,"toSeq":18234}`, records an `auditLog` `verify` entry with actor `system` and `"via":"cli"`, and exits with status 0 when intact, 1 when broken and 2 on error. Run it on a schedule (cron, your monitoring system) and alert on a non-zero exit; see Security Hardening.

10.3.2 What to do if verification fails

1. Do not prune, restore or otherwise change the database. Take a backup (see Backup and Restore) and keep it as evidence.
2. Note the entry id and reason. Look at the entries around it in the table.

3. Compare with your most recent backup: restore it to a separate instance and verify there to establish when the break appeared.
4. Treat it as a security incident: the change needed database owner rights or direct access to the database files.

10.4 Retention

The audit log is kept indefinitely unless an administrator prunes it. To prune:

1. In the **Retention** card, enter **Older than (days)**, 365 to 3650 (default 365).
2. Click **Prune...** and confirm **Prune audit entries?** with **Prune**.

The database removes the oldest run of the chain whose entries are all older than the cutoff, strictly in chain order, and records the last removed entry as the anchor that verification starts from. An old entry appended late, for example by a restore, waits until everything before it is old enough too. The database itself refuses to delete anything less than 365 days old, whatever the request.

Each prune writes an `auditLog` `delete` entry with the cutoff and the number removed, and emails every administrator ("Audit log entries pruned"). After a prune, verification reports chain positions from the anchor onward.

Before pruning, export or back up the period you remove if your retention policy requires you to keep it. Backups contain the audit log; see Backup and Restore for how a restore treats the live log.

10.5 Security alerts

Every active administrator is emailed (template `security.alert`, see Notifications and Email) when any of these happen:

Alert title	Raised when
Sign-in locked after repeated failures	A throttle lock is set
Repeated failed sign-ins for one account	An account reaches ten failures in 15 minutes
Administrator rights granted	An account is created as, or made, an administrator
Administrator rights removed	An administrator is demoted or deleted
Break-glass administrator signed in with a password	During SSO only
SSO-only sign-in turned off	
Audit log entries pruned	
Database restored from a backup	
Integration settings changed	SMTP, LDAP or SAML saved
API token created	
Session or API token limits changed	Settings → Sessions saved
Separation of duties setting changed	
AI assistant (MCP) access changed	
DPIA approval settings changed	
SCIM provisioning setting or token changed	
Administrator deactivated by SCIM provisioning	
Two-step verification reset for a user	An administrator resets a user's two-step verification
Two-step verification locked after repeated wrong codes	An account reaches 10 wrong code attempts in 15 minutes after a correct password
Two-step verification requirement changed	The requirement on Settings → Sessions is turned on or off

Alerts need SMTP to be configured. Each one is listed in the Notifications delivery log.

10.6 Evidence file fingerprints

Every attachment gets a SHA-256 fingerprint of its bytes at upload, stored on its row and recorded in the upload's audit entry. Files uploaded before fingerprinting existed are hashed by the worker: once when it starts and again in each nightly run, until every file present has one. It only reads files; a file missing from the uploads volume stays without a fingerprint.

On an evidence record, **Verify files** re-hashes each stored file and reports it as matching, changed since upload, or missing, with a `verify` audit entry. A changed or missing file means the uploads volume no longer holds what was uploaded: check the volume and the server backup (Backup and Restore). After restoring the uploads volume, Verify files on the evidence an assessor relies on confirms the files came back intact.

Locked evidence (USER-MANUAL, Locked evidence and file fingerprints) keeps its files, its dates and its links to completed control tests. The nightly run locks evidence that supports a recorded test result once its newest file is a day old. Unlocking needs `evidence:delete` at the `all` level and a reason, so in the built-in roles only GRC Manager and administrators can unlock; give that right sparingly.

10.7 Files moved onto evidence

Controls, compliance requirements and account reviews take evidence records instead of loose files (USER-MANUAL, Files on controls, requirements and account reviews). Files attached to them before that are moved by the worker when it starts, after the fingerprint backfill: each becomes an evidence record linked to the same record, typed "Other: Migrated from control" (or requirement, or account review), owned by the record's owner, with the file moved onto it unchanged. Each move is a `create` audit entry under the `evidence` module by `system`, naming the attachment and where it came from. A file whose record no longer exists stays where it is. Later starts find nothing to move. The people who could read those files now need `evidence:read` to see them; check custom roles before the update.

10.8 Using the audit log as evidence

The audit log supports common control requirements such as NIST SP 800-53 AU-2, AU-3, AU-9 and AU-10, ISO/IEC 27001 Annex A 8.15, and SOC 2 CC6 and CC7 change and access monitoring. When an auditor asks for evidence:

1. **Show integrity first.** Click **Verify audit log** (or run the command line check) and capture the "Intact" result with its chain positions and time. The `auditLog` `verify` entry in the log records the same.
2. **Show completeness of coverage.** Point to this chapter's list of what is recorded, and to the append-only and separate-role protections. Show a sample entry for each relevant category with **View diff**.
3. **Produce the population.** Filter to the category in scope and use **Export CSV**; the log is sorted newest first. For populations over 5,000 rows, export day by day with the **When** filter, or by

module, action or actor, or extract the period from the database (below).

4. **Show access to the log.** List who holds `auditLog:read` (the groups carrying GRC Manager, Auditor or a custom role with it) and the administrators. Only administrators can verify or prune.
5. **Show retention.** Filter Module `auditLog` , Action `delete` to show every prune and its cutoff, or show that there are none.

Typical evidence requests and how to answer them:

Request	Answer from the log
Access changes during the period	Module <code>users</code> : user, group and role entries with before and after
Privileged access granted	Module <code>users</code> , Entity <code>user</code> , entries where <code>isAdmin</code> changed; the matching "Administrator rights granted" alerts
Terminated users removed promptly	The <code>users</code> <code>update</code> setting <code>isActive</code> to false (or <code>scim.deprovision</code>), compared with the HR termination date
Failed and successful sign-ins	Module <code>auth</code>
Configuration changes	Module <code>settings</code>
Approvals and exceptions decided by the right person	The module's decision entries; self-approved decisions carry <code>selfApproved: true</code>
Evidence the log was not altered	The <code>verify</code> entries over the period, plus your external record of results

10.8.1 Keeping an external record

For the strongest evidence, record each verification outside TraxrGRC: run the command line check on a schedule, keep its JSON output (with the date) in a system the database owner cannot change, such as your SIEM or a write-once store. If the chain is ever rewritten, the recorded `toSeq` values and dates will not match what a later verification reports.

For extracting full entries with their diffs and hash values, the audit log is the `audit_log` table in the PostgreSQL database; a read-only query by an administrator (for example with `psql` in the `db` container) returns every column, including `chainSeq` , `prevHash` and `hash` . See Backup and Restore for database access.

Backup and Restore

TraxrGRC has two backup mechanisms. Use both; they protect against different failures.

	Server backup (<code>deploy/backup.sh</code>)	In-app snapshot (Settings → Backups)
Contents	A full PostgreSQL dump (every table, the audit log with its hash chain, the job queue) and an archive of the uploads volume (every uploaded file and the in-app snapshots).	Every application table as signed JSON, and with Include files in backups on (the default) every uploaded file beside it in a gzipped tar (<code>.tar.gz</code>).
Run by	The server operator, from cron.	An administrator in the browser, and the worker every Sunday at 03:00 UTC.
Stored	Wherever you point it, typically copied off the server.	Downloaded to the administrator's computer, or kept in the uploads volume (newest four).
Restores onto	A new, empty deployment, with <code>deploy/restore.sh</code> .	The running instance, replacing its data in place.
Use it for	Disaster recovery: lost server, lost disk, failed upgrade.	Rolling the data back to a known point, or moving data between instances that share a signing key.

The server backup is the one your recovery plan should rest on. It is the only mechanism that includes the job queue, and it does not depend on the application running.

11.1 Server backups

`deploy/backup.sh` backs up one instance. Run it from the installation directory:

```
cd /srv/grc
./deploy/backup.sh /var/backups/grc 14
```

The first argument is the target directory (created if missing). The second is how many days to keep (default 14). The script:

1. Finds the Compose project name the running stack uses (from `COMPOSE_PROJECT_NAME`, `.env` or the directory name), so it archives the uploads volume that the stack actually mounts. It stops if it cannot determine the name.
2. Runs `pg_dump` inside the `db` container as the database owner, in PostgreSQL's custom format, to `db-<stamp>.dump`. The dump is a consistent snapshot; the application can keep running.
3. Archives the uploads volume, read-only, through a temporary `alpine:3.20` container to `uploads-<stamp>.tar.gz`.
4. If `BACKUP_AGE_RECIPIENTS` is set, encrypts each file to `*.age` and removes the plain-text file.
5. Deletes backup files in the target directory older than the retention period.

`<stamp>` is the UTC time, for example `20261016T020000Z`. Files are created with mode 600. The script exits with an error and writes nothing further if any step fails, so a failed run is visible in its exit status and log.

The `db` container must be running, and `pg_dump` needs no other access. Run the script as an account that can use `docker`.

11.1.1 Scheduling

Run the backup nightly from cron, for example at 02:00:

```
0 2 * * * cd /srv/grc && ./deploy/backup.sh /var/backups/grc 14 >> /var/log/grc-backup.log 2>
```

Cron runs with a short `PATH`. If `docker` (or `age`) is not in `/usr/bin` or `/bin` on your server, set `PATH` at the top of the crontab. Check the log after the first scheduled run, and alert on a missing or failed run as you would for any other backup job.

11.1.2 Encrypting backups

To make the off-site copies unreadable without a key that never needs to be on the server, encrypt them with `age`:

1. On a machine other than the server, create a key pair:

```
age-keygen -o grc-backup-key.txt
```

The command prints the public key (`age1...`). Store `grc-backup-key.txt`, the private key, in your password vault. Without it the backups cannot be restored.

2. On the server, put the public key in a file, for example `/srv/grc/backup-recipient.txt`. List that file in `.deploy-keep` so upgrades leave it alone (see Upgrades).
3. Install `age` on the server and run the backup with the recipient file:

```
BACKUP_AGE_RECIPIENTS=/srv/grc/backup-recipient.txt ./deploy/backup.sh /var/backups/grc
```

The script encrypts as a separate step after each file is complete, so a failure in either step stops the run.

11.1.3 Copying backups off the server

A backup on the same disk as the database does not survive losing that disk or the server. Copy the backup directory to another machine or to object storage after each run, with the tool your organization already uses (rsync, restic, a cloud storage client). Keep the copy's retention at least as long as the local one, and restrict who can read it: an unencrypted dump holds every record, password hashes and the encrypted integration secrets.

11.1.4 What else to keep

A restore needs more than the backup files:

- `.env`, in particular `AUTH_SECRET`, `POSTGRES_PASSWORD`, `APP_DB_PASSWORD` and, if set, `BACKUP_SIGNING_KEY` and `SETTINGS_ENCRYPTION_KEY`. Keep it in the vault.
- **The release** the backup was taken with, or a later one.
- **The age private key**, if backups are encrypted.
- Any `Caddyfile.*`, `docker-compose.override.yml` and certificates you added.

11.2 Restoring a server backup

`deploy/restore.sh` restores one backup onto a **new, empty** deployment. It refuses to run over a database that already has tables.

1. Prepare the target: unpack the release into a new directory, put the original `.env` there, and do not start the stack.
2. Copy the two backup files from the same run (`db-<stamp>.dump` and `uploads-<stamp>.tar.gz`, or their `.age` versions) to the server.
3. Run the restore from the new installation directory:

```
./deploy/restore.sh /var/backups/grc/db-<stamp>.dump /var/backups/grc/uploads-<stamp>.tar.gz
```

For encrypted files, pass the `.age` files and the private key:

```
BACKUP_AGE_IDENTITY=/root/grc-backup-key.txt ./deploy/restore.sh \  
/var/backups/grc/db-<stamp>.dump.age /var/backups/grc/uploads-<stamp>.tar.gz.age
```

4. Start the stack:

```
docker compose --profile tls up -d --build
```

5. Verify the result as in steps 6 to 9 of the restore drill below.

The script:

1. Checks that both files exist and finds the project's uploads volume the same way `backup.sh` does.
2. For `.age` files, requires `BACKUP_AGE_IDENTITY`, decrypts into a temporary directory and removes it when done.
3. Starts only the `db` service and waits for it to be healthy.
4. Counts the tables in the database and stops with `The database already has <n> tables. Restore only onto a new, empty deployment.` if there are any.
5. Restores the dump as the owner with `pg_restore --no-owner --no-privileges --exit-on-error`. Grants are left out on purpose: the application role does not exist yet, and `migrate` creates it and reapplies its grants on the first start.
6. Unpacks the uploads archive into the project's uploads volume.

The audit log, its hash chain and the prune anchors are restored exactly as they were, so the restored trail verifies as the original did. Users keep their passwords. Sessions do not carry over if `AUTH_SECRET` differs.

Keep `AUTH_SECRET` (and `SETTINGS_ENCRYPTION_KEY`, if set) the same as on the original instance. With a different key, everyone has to sign in again and the SMTP password, LDAP bind password and SAML private key read as empty until an administrator enters them again under **Settings → Integrations**.

11.2.1 Restoring in place on the same server

`restore.sh` never overwrites data, so to restore over a damaged instance on the same server you first remove its volumes. This destroys the current data: take a backup of what is there first, even if it is damaged.

```
cd /srv/grc
./deploy/backup.sh /var/backups/grc-before-restore 3650
docker compose --profile tls down
docker volume rm grc_pgdata18 grc_uploads
./deploy/restore.sh /var/backups/grc/db-<stamp>.dump /var/backups/grc/uploads-<stamp>.tar.gz
docker compose --profile tls up -d
```

Replace `grc` in the volume names with your project name. The uploads volume must be removed too: the archive is unpacked over whatever the volume holds.

11.3 In-app snapshots

Settings → Backups exports and restores signed snapshots of the database, with the uploaded files when **Include files in backups** is on. Open it from the **Backups** card on the Settings page (address `/settings/backups`). Only administrators can open it, because snapshots hold password hashes and the encrypted integration secrets.

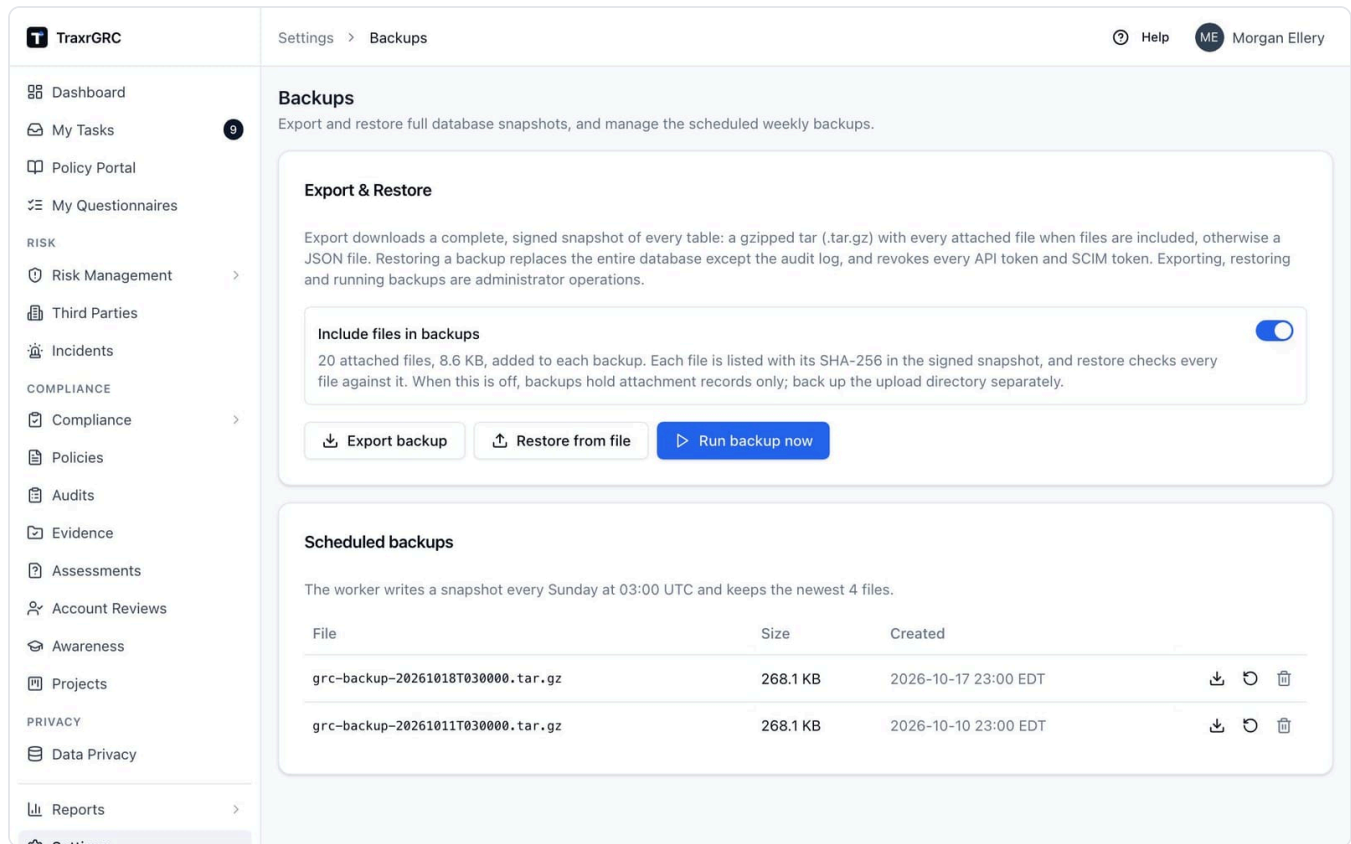


Figure 11.1. The Backups screen with the Export & Restore card, the Include files in backups switch and two scheduled backups, each with Download, Restore and Delete buttons.

11.3.1 What a snapshot contains

A snapshot holds every row of every application table, read in a single consistent transaction, with a signature. It leaves out upload links for AI assistants (short-lived credentials) and the notification rules' read position in the audit log.

Include files in backups (on the Backups screen, on by default) decides whether the files come too, for every kind of record: evidence, policies, risks, vendors and the rest.

- **On:** the backup is a gzipped tar (`.tar.gz`) holding the signed snapshot as its first entry, `snapshot.json` , and every attached file under `files/` . The snapshot lists each file with its SHA-256 and size, so the signature covers the hashes. A file whose record exists but which was already missing from the uploads volume is listed as missing. The screen shows how many files

there are and how much they add to each backup. There is no size limit beyond free disk space (see Where backups are assembled).

- **Off:** the backup is the signed JSON file alone. Attachment records are included, but the files themselves are only in the uploads volume and the server backup.

Changing the setting is recorded in the audit log. The file name is `grc-backup-YYYY-MM-DD.tar.gz` (or `.json`) for an export and `grc-backup-YYYYMMDDTHHMMSS.tar.gz` (or `.json`) for a scheduled or on-demand snapshot. Backups with files made by earlier versions are `.zip` files; they still restore.

The database part of a snapshot is read back whole during a restore, so it can be at most 500 MB of JSON. Attached files do not count toward that.

11.3.2 Where backups are assembled

Exports, scheduled snapshots and restores work on disk, never holding a whole backup in memory. Each job gets its own private directory under `BACKUP_WORK_DIR` (by default `backup-work` inside the uploads volume), which is removed when the job ends; the worker also removes any left by a crash once they are a day old.

- An export streams the tables to disk one at a time, hashes each attached file, signs the snapshot and streams the snapshot and every file into the archive. The download is sent from that file.
- A restore unpacks the archive into its directory, checking each file against the signed snapshot, before the database is touched.

Before starting, both check that the volume has room for the job plus 256 MB to spare, and refuse with **Not enough free disk space** and the amounts when it does not. A disk that fills up mid-job ends it with the same message and leaves the data as it was. Plan for free space of about twice the size of the attached files: one copy for the files themselves and one for the backup being assembled or unpacked.

11.3.3 Signing

Every snapshot is signed with HMAC-SHA256. The key is derived (HKDF-SHA256) from `BACKUP_SIGNING_KEY` when it is set, otherwise from `AUTH_SECRET`, with a label of its own so it is unrelated to the session keys. A restore checks the signature:

File	Result
Signed by this instance's key, unchanged	Restored. The audit log records the signature as verified.
Edited after export, or signed with a different key	Refused.
No signature (exported before signing existed)	Refused until the administrator ticks I trust this unsigned file . The restore is recorded as unsigned and the file's audit entries are not imported.

Warning: Changing `AUTH_SECRET` without a `BACKUP_SIGNING_KEY` makes earlier snapshots unrestoreable in the application.

To restore one instance's snapshots on another, give both the same `BACKUP_SIGNING_KEY`. Server backups are not affected by any of this.

11.3.4 Exporting and the scheduled snapshots

- **Export backup** downloads a fresh snapshot to your computer.
- **Run backup now** writes a snapshot into the uploads volume and adds it to **Scheduled backups**.
- The worker writes the same kind of snapshot every Sunday at 03:00 UTC. The newest four are kept; older ones are deleted from the list and the disk.
- Each row in **Scheduled backups** has a download action, a restore action and a delete action. Deleting asks for confirmation in **Delete backup file?**; restoring opens the same **Restore backup?** dialog as a file you upload.

Exports, on-demand runs and deletions are recorded in the audit log; an export with files also records how many files it carried and their size. A failed scheduled snapshot is logged by the worker and recorded in the audit log as `backup.failed`.

The scheduled snapshots live on the same volume as the data they protect, so they do not replace the server backup. They are copied off the server as part of the uploads archive.

11.3.5 Restoring a snapshot

1. Click **Restore from file** and choose a backup file: `.tar.gz`, `.json`, or a `.zip` from an earlier version. To restore one of the **Scheduled backups** instead, click the restore action on its row; the file never leaves the server, which suits a large backup.
2. The **Restore backup?** dialog names the file. Type `RESTORE` in **Type RESTORE to confirm**.
3. If the file is unsigned, a warning appears; tick **I trust this unsigned file** only if you know where the file came from.

4. Click **Restore**. When it finishes, sign in again.

The restore runs in one database transaction, so it either completes or changes nothing. It:

- Deletes and recreates every table from the file, except four whose live rows are kept: the audit log, the list of scheduled snapshot files, the SAML replay records and the sign-in lockouts.
- Adds the snapshot's audit entries that the live audit log does not already have, for signed files only. Live audit entries are never changed.
- Revokes every API token and every SCIM token. Issue new tokens to every integration and to your identity provider's provisioning afterward.
- Records the restore in the audit log with the record counts before and after, and emails every administrator a security alert.

Restoring a backup with files also brings its files back. The archive is unpacked into the job's directory first and each file is checked against the SHA-256 the signed snapshot lists; the verified files move into the uploads volume once the database transaction has committed, so a refused or failed restore leaves the uploads volume untouched. The archive must start with the signed snapshot, and it is refused outright if it holds anything the snapshot does not list (including links, devices and paths outside `files/`) or more data than the snapshot declares. Within that:

- A file that matches is written back, or left alone when the volume already holds the same bytes.
- A file that does not match is not written. The restore reports it by name, and the audit entry lists it.
- A listed file the archive does not carry is reported as missing.

A backup with files must carry this instance's signature, since only the signature vouches for the hashes; an unsigned one is refused. Restoring a JSON snapshot does not touch the uploads volume: attachment records point at files by their stored names, and files that are not in the volume can no longer be downloaded.

11.4 Restore drill

Test the server backup regularly (at least quarterly, and after any change to the backup job) by restoring it onto a throwaway instance on the same server. The drill below leaves production untouched and sends no email.

1. On production, record the row counts and take a backup straight after:

```
cd /srv/grc
docker compose exec -T db sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -At -F " " -
  < deploy/counts.sql | sort > /tmp/counts-prod.txt
./deploy/backup.sh /var/backups/grc-drill 1'
```

2. Create the drill directory from the same release and a copy of `.env`:

```
mkdir /srv/grc-drill
tar -xzf grc-release-<version>.tar.gz -C /srv/grc-drill
cp /srv/grc/.env /srv/grc-drill/.env
```

3. Edit `/srv/grc-drill/.env`:

```
COMPOSE_PROJECT_NAME=grc-drill
APP_IMAGE=grc-platform:drill
APP_URL=http://localhost:3100
WEB_PORT=3100
```

Keep `AUTH_SECRET` and the other keys as they are, so the restored integration secrets stay readable.

4. Restore into the drill project:

```
cd /srv/grc-drill
./deploy/restore.sh /var/backups/grc-drill/db-<stamp>.dump /var/backups/grc-drill/upload
```

5. Start only `db`, `migrate` and `web`. Leaving out `worker` and `caddy` means the drill instance cannot send email (only the worker does) and does not compete for ports 80 and 443:

```
docker compose up -d --build web
```

6. Check that `migrate` found nothing to apply and skipped the initial administrator:

```
docker compose logs migrate | grep -E 'No pending migrations|users exist'
```

7. Compare the row counts with production. Only tables whose counts differ are printed; expect none, or only small differences in tables production wrote to between step 1 and the backup:

```
docker compose exec -T db sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -At -F " " -
    < deploy/counts.sql | sort > /tmp/counts-drill.txt
join -a 1 -a 2 -e MISSING -o 0,1.2,2.2 /tmp/counts-prod.txt /tmp/counts-drill.txt \
    | awk '$1 ~ /^public\.\/ && $2 != $3'
```

8. Verify the audit chain:

```
docker compose exec web ./node_modules/.bin/tsx prisma/audit-verify.ts
```

Expect `{"ok":true,...}`, with `rows` equal to the drill's `public.audit_log` count from step 7.

9. Sign in through an SSH tunnel (`ssh -L 3100:127.0.0.1:3100 <server>`), then open `http://localhost:3100`, open a record with an attachment and download it.

10. Remove the drill instance, its volumes and its image:

```
docker compose down -v
docker image rm grc-platform:drill
cd / && rm -rf /srv/grc-drill /var/backups/grc-drill /tmp/counts-prod.txt /tmp/counts-di
```

`down -v` deletes the drill's volumes. Run it only in `/srv/grc-drill`, never in the production directory.

Record the date, the backup used, and the results of steps 6 to 9 as evidence of a tested restore.

Upgrades

Upgrade an instance with `deploy/update.sh`. The script backs up the instance, installs the new release, rebuilds, restarts and then proves the instance came up with its data intact. Run it for every upgrade.

12.1 Before you upgrade

- Read the release notes for manual steps.
- Make sure the last nightly backup succeeded and its off-site copy exists.
- Check free disk space: the update writes a new backup and builds a new image.
- Plan a short maintenance window. Users stay signed in, but the application is unavailable while the containers are recreated, typically for a minute or two after the build.

12.2 Running an update

1. Unpack the new release into its own directory, never over the installed one:

```
mkdir /tmp/grc-release-<version>
tar -xzf grc-release-<version>.tar.gz -C /tmp/grc-release-<version>
```

2. From the installation directory, run the update script **from the new release**, so that fixes to the script itself apply:

```
cd /srv/grc
/tmp/grc-release-<version>/deploy/update.sh /tmp/grc-release-<version> <version> --prof:
```

The arguments are:

Argument	Meaning
<code><release-dir></code>	The unpacked new release. It must contain <code>docker-compose.yml</code> and must not be the installation directory.
<code><version></code>	The version label recorded in <code>.deployed-commit</code> and used to name the next rollback image.
<code>--profile NAME</code>	A Compose profile to build and start with. Repeat it for several profiles. Use <code>--profile tls</code> with the bundled Caddy.
<code>--backup-dir DIR</code>	Where the backup and row counts go. Default <code>~/grc-deploy-backups/<project></code> .

3. Read the output. A successful run ends with `update: <project> is at <version>`.

4. Delete the unpacked release: `rm -rf /tmp/grc-release-<version>`.

12.3 What the update does

The script runs these steps in order and stops at the first failure.

12.3.1 Plan the file changes

The image is built from the whole installation directory, so a file left behind by an earlier release (a removed page, say) would still be built and served. The script therefore deletes files that the new release no longer contains. To make that safe, it may delete only files that the previous update installed, which it recorded in `.deployed-files`.

Any other file in the installation directory that is not in the release stops the update before anything changes, with exit status 3:

```
update: refusing: these files are here, not in the release, and were not installed by the last
backup-to-offsite.sh
Remove them by hand, or list them in .deploy-keep if they belong to this machine. Nothing was
```

Remove the file, or, if it belongs to this server, list it in `.deploy-keep` (one path per line, relative to the installation directory, `#` comments allowed) and run the update again.

These are never deleted or overwritten, whether or not you list them:

- `.env`, `.deploy-keep`, `.deployed-commit`, `.deployed-files`
- `.initial-admin`, `.demo-accounts`
- `docker-compose.override.yml`

- `Caddyfile.*` at the top level
- the `node_modules/`, `.next/` and `uploads/` directories
- every path listed in `.deploy-keep`

The first update of an installation that has no `.deployed-files` yet refuses any deletion, because it cannot tell installed files from your own. Check the list it prints, remove what is stale, and run it again.

When the plan is accepted, the script prints a summary such as `update: grc 1.4.0 -> 1.5.0: 1840 files, 2 to delete` and the files it will delete.

12.3.2 Back up and count rows

The script runs the new release's `deploy/backup.sh` into the backup directory with 30 days' retention, then runs `deploy/counts.sql` to count the rows of every table in the application and queue schemas, saving the result as `counts-before-<stamp>.txt` in the same directory.

12.3.3 Tag a rollback image

If `web` is running, the script tags the image the running container uses as `<image>:rollback-<previous version>`, where `<image>` is the web image name the new release's `docker-compose.yml` configures (by default `grc-platform-<COMPOSE_PROJECT_NAME>`), for example `grc-platform-grc:rollback-1.4.0`. It tags the running container's image by its id, so the tag holds the code that is live now even if the image name has changed. The previous version is read from `.deployed-commit` (`none` on an installation that has never been updated with the script).

12.3.4 Install the release

The release's files are copied into the installation directory, skipping the protected files above, and the stale files found by the plan are deleted, along with any directories they leave empty.

12.3.5 Build and start

The script runs `docker compose build` and `docker compose up -d` with your profiles. `migrate` applies any new database migrations, refreshes the application role's grants and runs the bootstrap before `web` and `worker` restart. The script then waits for `web` to report healthy, checking every `UPDATE_POLL_SECONDS` seconds (default 3) up to 60 times.

12.3.6 Verify and record

1. Runs `prisma/audit-verify.ts` in the `web` container and fails unless the audit chain verifies.
2. Counts rows again (`counts-after-<stamp>.txt`) and fails if any table in the application schema lost rows or disappeared. Tables that gained rows are listed for information. Queue tables are not compared; finished jobs are cleaned up there normally.
3. Calls `/api/health` inside the `web` container.

4. Writes the version to `.deployed-commit` and the release's file list to `.deployed-files`.

`.deployed-commit` and `.deployed-files` are written only when every check passes, so the version on record is always the last update that verified.

12.4 When an update fails

Fails at	State of the instance	What to do
Usage or plan (exit status 2 or 3)	Unchanged.	Correct the arguments or the unexpected files and run again.
Backup or row count	Unchanged. The files have not been installed.	Fix the cause (usually disk space or a stopped <code>db</code>) and run again.
Build, start, health, audit chain, row counts	The new release's files are installed; the database may have been migrated.	Read the failure, then fix forward or roll back (below).

After the files have changed, a failure prints the backup directory, the stamp, and the command that returns to the previous image:

```
update: FAILED after the files changed: web did not become healthy (last status: unhealthy)
Backup and row counts: /home/grc/grc-deploy-backups/grc (stamp 20261016T020000Z).
To return to the previous image: docker tag grc-platform-grc:rollback-1.4.0 grc-platform-grc:
```

Start with the logs: `docker compose logs migrate` for a failed start, and `docker compose logs web` for a health failure (see Monitoring and Troubleshooting).

12.5 Rolling back

Choose the rollback by what the failed release changed in the database.

Image rollback. If `migrate` applied no new migrations (its log says `No pending migrations to apply.`), the previous image runs against the database as it is. Run the command the script printed; it names this instance's own image:

```
docker tag grc-platform-grc:rollback-1.4.0 grc-platform-grc:latest
docker compose --profile tls up -d
```

The installation directory still holds the new release's files, so a later `docker compose build` or `up --build` would build the new release again. Before the next build, run the previous release's `update.sh`

with the previous release directory, or the fixed new release.

Full rollback. If the failed release applied migrations, the previous image may not work with the changed database. Return both code and data to the moment before the update:

1. Stop the application and keep the current state for investigation:

```
docker compose stop web worker
./deploy/backup.sh /var/backups/grc-failed-update 3650
docker compose --profile tls down
```

2. Restore the backup the update took before installing (`db-<stamp>.dump` and `uploads-<stamp>.tar.gz` in the backup directory) onto a new directory holding the previous release and the same `.env`, as described in Restoring a server backup. To reuse the installation directory and project name, remove the volumes first as in Restoring in place on the same server.
3. Start the previous release and verify it as in the restore drill.

Warning: Data entered between the update and the rollback is lost in a full rollback. Recover it from the backup taken in step 1 if needed.

12.6 Cleaning up after upgrades

Each update leaves a rollback image and a backup. Once a release has run well for a while (replace `grc-platform-grc` with your instance's image name, `grc-platform-<COMPOSE_PROJECT_NAME>` unless you set `APP_IMAGE`):

```
docker image ls grc-platform-grc
docker image rm grc-platform-grc:rollback-<old version>
docker image prune
docker builder prune
```

The update's backups in `~/grc-deploy-backups/<project>` are pruned after 30 days by the next update. Remove older ones by hand if updates are infrequent.

12.7 Upgrading an installation from before October 2026

An installation started before 15 October 2026 needs one-time steps when it first moves to a newer release. Releases from 14 October 2026 connect `web` and `worker` as the separate application role: add `APP_DB_PASSWORD` to `.env` (Compose refuses to start without it) and upgrade as usual. Releases from 15 October 2026 run PostgreSQL 18 on a new volume, `pgdata18`, and PostgreSQL 18 cannot

open the older data directory: move the data by taking a backup with the old release, restoring it with `deploy/restore.sh` onto the new volume, and starting the new release. The old `pgdata` volume is left untouched for rollback. Follow the release notes for these versions step by step.

Warning: Skipping the PostgreSQL 18 data move starts the new release with an empty database. Your data is still in the old `pgdata` volume, but anything entered in the empty instance is not.

Monitoring and Troubleshooting

This chapter covers what to watch on a running instance, where to find logs, and how to fix the failures operators meet most often.

13.1 Health checks

13.1.1 The health endpoint

`GET /api/health` needs no sign-in and reports whether the application can reach its database:

Response	Meaning
<code>200</code> with <code>{"status":"ok"}</code>	<code>web</code> is running and the database answered.
<code>503</code> with <code>{"status":"unavailable"}</code>	<code>web</code> is running but the database query failed.
No response, or a proxy error	<code>web</code> is down or unreachable.

The endpoint reveals nothing else and is never cached. Point your external monitoring at the public address, for example `https://grc.example.com/api/health`, so it also covers DNS, the proxy and the certificate.

13.1.2 Container health checks

Service	Check	Healthy when
<code>db</code>	<code>pg_isready</code> every 5 seconds	The database accepts connections. Up to 20 retries.
<code>web</code>	<code>wget</code> of <code>http://127.0.0.1:3000/api/health</code> every 15 seconds	The endpoint answers <code>200</code> . Unhealthy after 5 failures; failures in the first 20 seconds do not count.
<code>worker</code>	None	Check that it is running and read its log.
<code>migrate</code>	None	It must exit with code 0.

`docker compose ps` shows the state of each service. `restart: unless-stopped` restarts `db`, `web`, `worker` and `caddy` if they exit. Docker does not restart a container that is only unhealthy.

13.1.3 The audit chain

Verify the audit log's hash chain daily from cron, so a break is reported rather than found later:

```
30 3 * * * cd /srv/grc && docker compose exec -T web ./node_modules/.bin/tsx prisma/audit-ver
```

The command prints the verdict as JSON and exits 0 when the chain is intact, 1 when it is broken and 2 on an error. Each run is itself recorded in the audit log. Administrators can run the same check under **Settings → Audit Log** (`/settings/audit-log`); see Audit Log and Evidence Integrity.

To detect a rewrite by someone holding the database owner's password, also copy the chain head off the server after each verification:

```
docker compose exec -T db sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -tAc "SELECT \'la
```

A later rewrite no longer matches a hash recorded elsewhere.

13.1.4 What else to watch

Watch	How	Why
Nightly backup	Exit status and log of the cron job; age of the newest file off-site	The backup is your recovery.
Disk space	<code>df -h</code> on the Docker data directory and the backup directory	A full disk stops the database.
Security alerts	Administrators' mailboxes	Lockouts, break-glass sign-ins and administrator changes are reported only by email.
Email delivery	Settings → Notifications delivery list; <code>[mailer]</code> lines in the worker log	Failed deliveries mean reminders and alerts are not arriving.
Certificate expiry	External monitoring of the HTTPS address	With your own certificate, renewal is yours to do.

13.2 Logs

Each service writes to standard output; read it with `docker compose logs`:

```
docker compose logs --tail 200 web
docker compose logs -f worker
docker compose logs migrate
docker compose logs caddy
docker compose logs db
```

Service	What to look for
migrate	Prisma's migration output, then <code>app-role:</code> and <code>[bootstrap]</code> lines. An error here is why a start stopped.
web	Request errors, and structured JSON lines with an <code>event</code> field, such as <code>settings.decrypt_failed</code> and <code>security_alert.failed</code> .
worker	One <code>[worker]</code> line when it is ready, then a line per daily sweep, weekly backup and scheduled report run; <code>[mailer] delivery <id> failed:</code> for each failed email.
caddy	Certificate requests and renewals, and upstream errors when <code>web</code> is down.
db	Connection and authentication errors, disk errors.

Docker's default log driver keeps container logs without limit. Set a size limit in `/etc/docker/daemon.json`, for example:

```
{
  "log-driver": "json-file",
  "log-opts": { "max-size": "50m", "max-file": "5" }
}
```

Restart Docker and recreate the containers for it to apply. If your organization collects logs centrally, ship the container logs there; the audit log inside the application remains the record of user activity.

13.3 Row counts

`deploy/counts.sql` prints the row count of every table in the application and queue schemas, one `schema.table count` line each. Take it before and after any maintenance you do by hand, and compare:

```
docker compose exec -T db sh -c 'psql -U "$POSTGRES_USER" -d "$POSTGRES_DB" -At -F " " -f -'
< deploy/counts.sql | sort > counts-$(date -u +%Y%m%dT%H%M%S).txt
```

13.4 Common problems

13.4.1 The stack does not start: a variable is not set

Compose prints `Set APP_DB_PASSWORD in .env (or AUTH_SECRET, APP_URL, POSTGRES_PASSWORD )` and starts nothing. Add the value to `.env` and run the command again.

13.4.2 migrate fails

`web` and `worker` stay down until `migrate` succeeds. Read its log:

```
docker compose logs migrate
```

Message	Cause and fix
<code>[bootstrap] No users exist. Set INITIAL_ADMIN_EMAIL and INITIAL_ADMIN_PASSWORD ...</code>	A new database and no initial administrator. Set both in <code>.env</code> (password 12 characters or more) and start again.
<code>app-role: APP_DB_PASSWORD must be set, 16 characters or more.</code>	Set a longer <code>APP_DB_PASSWORD</code> .
<code>app-role: APP_DB_USER must differ from the database owner ... or must be a lower-case PostgreSQL identifier</code>	Correct <code>APP_DB_USER</code> .
<code>app-role: <role> owns <tables> in public; it must own nothing there.</code>	Tables were created by the application role, usually by hand. Transfer their ownership to the owner role and start again.
An authentication error for the owner role (<code>grc</code>)	<code>POSTGRES_PASSWORD</code> in <code>.env</code> differs from the password the database volume was created with. The postgres image sets the password only when it creates the volume; restore the original value.
A Prisma migration error (<code>P3009</code> , <code>P3018</code>)	A migration failed part way. Do not edit the database by hand. Keep the volume, take a backup, and contact your support channel with the log; restore the pre-update backup if you need the service back first (see Upgrades).

To inspect migration state without starting the application, run a command in a one-off `migrate` container:

```
docker compose run --rm --no-deps migrate ./node_modules/.bin/prisma migrate status
```

13.4.3 web is unhealthy or keeps restarting

```
docker compose ps web
docker compose logs --tail 100 web
curl -s http://127.0.0.1:3000/api/health
```

Symptom	Cause and fix
Log shows <code>AUTH_SECRET</code> is shorter than 32 characters or <code>AUTH_SECRET</code> is still the <code>.env.example</code> placeholder	In production the application refuses a weak session secret. Generate one with <code>openssl rand -base64 32</code> , set it, and run <code>docker compose up -d</code> . Changing it signs everyone out (see Security Hardening).
<code>/api/health</code> returns <code>{"status":"unavailable"}</code>	<code>web</code> cannot reach the database. Check <code>docker compose ps db</code> and <code>docker compose logs db</code> ; check that <code>APP_DB_PASSWORD</code> has not changed since <code>migrate</code> last ran (<code>docker compose up -d</code> reapplies it).
Health is fine locally but the site does not load	The proxy is the problem: <code>docker compose logs caddy</code> , or your proxy's log. Check DNS for <code>DOMAIN</code> and that ports 80 and 443 are open.
Sign-in redirects to the wrong address	<code>APP_URL</code> is wrong. Set it to the public address and run <code>docker compose up -d</code> .
Form submissions or uploads fail with 403 behind your own proxy	The proxy does not pass the browser's <code>Host</code> header (or <code>X-Forwarded-Host</code>), so the origin check fails. Pass it.
Uploads fail with 413	The proxy's body limit is lower than the application's. Use 32 MiB, and 2049 MiB for <code>/api/backups/restore</code> .

13.4.4 Caddy does not get a certificate

`docker compose logs caddy` shows the ACME error. Check that `DOMAIN` resolves to this server from the internet, that ports 80 and 443 reach Caddy (no other service holds them, and no firewall blocks

them), and that `DOMAIN` and `ACME_EMAIL` are set. For a server Let's Encrypt cannot reach, use your own certificate (see Installation).

13.4.5 Email is not sent

1. In **Settings → Integrations**, click **Send test email**. The error it shows is the SMTP server's answer.
2. If the test works but notifications do not arrive, check that the worker is running (`docker compose ps worker`) and read its log for `[mailer] delivery <id> failed:` lines. Email is sent only by the worker.
3. A configuration that sends a username and password requires STARTTLS (or implicit TLS) from the relay; a relay that does not offer it is refused rather than sent the password in clear.
4. If the web log shows `settings.decrypt_failed` for `smtp.pass`, the stored password was encrypted with a different key (after a change of `AUTH_SECRET` or `SETTINGS_ENCRYPTION_KEY`, or a restore from another instance). Enter the password again under Settings → Integrations.
5. Digests arriving at the wrong time after a schedule change: the cron is in UTC. If saving the schedule warned that it takes effect when the worker next starts, restart the worker (`docker compose restart worker`).

Notification setup and the delivery list are described in Notifications and Email.

13.4.6 Single sign-on errors

A refused LDAP or SAML sign-in shows the user a general message and records the reason in the audit log. Look up the sign-in failure entry under Settings → Audit Log, and see Authentication and Single Sign-On for each reason. On the server side, check that `APP_URL` uses `https://` and matches the address registered with the identity provider, and that the server clock is correct: SAML responses older than 10 minutes are refused.

13.4.7 Disk full

A full disk stops PostgreSQL, and with it the application. Find what is using the space:

```
df -h
docker system df
du -sh /var/backups/grc
```

Free space by, in order of safety:

1. Moving older local backups off the server, or shortening the local retention period.
2. Removing old rollback images and the build cache (see Upgrades, Cleaning up after upgrades).
3. Limiting container log size (see Logs).
4. Deleting in-app snapshots under Settings → Backups, if they are copied elsewhere.

Never remove a Docker volume to free space. After space is freed, run `docker compose --profile tls up -d` and check health.

13.4.8 Audit chain verification fails

A broken chain is evidence that an audit entry was changed, deleted or reordered outside the application, which requires the database owner's password or root on the server. The verdict names the first entry that fails and the reason:

Reason	Meaning
<code>hash-mismatch</code>	The entry's content no longer matches its stored hash: it was changed.
<code>prev-hash-mismatch</code>	The entry does not follow the entry before it: an entry was removed or inserted.
<code>sequence-gap</code>	A sequence number is missing: an entry was deleted.
<code>unchained</code>	An entry has no chain fields.
<code>before-anchor</code>	An entry older than the last retention prune is still present.
<code>head-mismatch</code>	The last entry does not match the recorded chain head.

Do this:

1. Do not repair anything, and stop further writes if you can (`docker compose stop web worker`).
2. Take a server backup into a separate directory and preserve the database volume as it is.
3. Record the verdict and the time, and start your incident process. Compare with the chain heads you copied off the server to find when it changed.
4. Restoring an earlier backup brings back an intact trail from that time; decide that as part of the investigation, after the evidence is kept.

13.4.9 Locked out of administrator accounts

Sign in with a break-glass administrator; it keeps password sign-in when **SSO only** is on and the identity provider is down. Any administrator can reset another user's password under Settings → Users. Repeated failed sign-ins block the address that failed, never the account, so the right password from another network still works. There is no command-line password reset, which is why the post-installation checklist asks for a second break-glass administrator.

Security Hardening

This chapter lists what TraxrGRC enforces by itself and what you, as the operator, must do. Work through the operator checklist before go-live and again after each major change.

14.1 What the product enforces

Area	Built-in behavior
Transport headers	Every response carries <code>Strict-Transport-Security: max-age=31536000; includeSubDomains</code> , <code>X-Content-Type-Options: nosniff</code> , <code>X-Frame-Options: DENY</code> , <code>Referrer-Policy: strict-origin-when-cross-origin</code> and a <code>Permissions-Policy</code> that turns off camera, microphone, geolocation and payment. The <code>X-Powered-By</code> header is not sent.
Content Security Policy	Pages carry a per-request nonce policy: scripts only from the application with the request's nonce, no plugins, no framing (<code>frame-ancestors 'none'</code>), forms and connections to the application only.
Cross-site requests	Server actions and every browser route that changes data refuse requests whose <code>Origin</code> does not match the host.
Session secret	<code>web</code> refuses to start in production with a missing, short (under 32 characters) or placeholder <code>AUTH_SECRET</code> .
Sessions	Sessions end after the session length (default 12 hours) or the idle timeout (default 60 minutes). Rights are refreshed every minute; a password reset, two-step verification reset, deactivation, administrator or group change, or Sign out everywhere ends the user's sessions within a minute, and a user's own password change ends their other sessions.
Sign-in throttling	Five failures lock that email from that address for 15 minutes; 30 failures lock the address for every account. Accounts are never locked outright. Ten failures on one account in 15 minutes refuse every address that failed and alert administrators.
Directory and SSO	LDAP requires <code>ldaps://</code> or StartTLS. SAML responses must be signed, answer a sign-in started from the application within 10 minutes, and name this instance as recipient; each is accepted once. LDAP and SSO never take over an existing password, administrator or break-glass account unless an administrator allowed the link.
Multi-factor authentication	Built-in two-step verification (TOTP authenticator apps, with ten single-use recovery codes) for password and LDAP sign-in, optional per user or required for everyone under Settings → Sessions . Single sign-on relies on the identity provider's MFA; SSO only makes it the only way in, with break-glass administrators keeping password sign-in for outages.
API rate limit	Each API token may make 600 requests a minute on <code>/api/v1</code> by default (<code>API_RATE_LIMIT_PER_MINUTE</code>); further requests get <code>429</code> with <code>Retry-After</code> .

Area	Built-in behavior
Stored secrets	The SMTP password, LDAP bind password, SAML private key and two-step verification secrets are encrypted in the database (AES-256-GCM). Recovery codes are stored as keyed HMAC-SHA256 hashes. Passwords are stored as bcrypt hashes. API, SCIM and upload-link tokens are stored as SHA-256 hashes.
Uploads	Attachments are limited to 25 MB each and to an allowlist of types (PDF, common images, text, Markdown, CSV, Office documents and ZIP), checked by extension, declared type and file content. Files are stored under random names and downloaded as attachments. Each user has an attachment storage quota (default 2048 MB).
Audit log	Append-only for the application: the application's database role can only read and add entries, triggers refuse changes, and a hash chain makes any later change detectable. Entries are kept at least 365 days.
Backups	In-app snapshots are signed and checked on restore; restore keeps the audit log and security state and revokes all API and SCIM tokens.
Containers	Non-root, read-only root filesystem, all capabilities dropped, <code>no-new-privileges</code> ; <code>web</code> published on loopback only; the database never published.
Outbound calls	Telemetry and update checks are off. The application connects only to the services you configure (SMTP, LDAP, Microsoft Entra ID for AI assistants).

14.2 Operator checklist

14.2.1 TLS and the network

- Serve the instance over HTTPS only, through the bundled Caddy or your own proxy.
- Give the instance a host name of its own. HSTS applies to the host name on every port and to every name below it, for a year; anything else served on that name must also use HTTPS. The header deliberately omits `preload`; submitting the domain to browser preload lists is the domain owner's decision.
- Leave `WEB_BIND` at `127.0.0.1` unless a proxy on another host needs it, and then firewall port 3000 to that proxy. A client that reaches port 3000 directly can choose the address recorded for it in throttling and the audit log.
- With your own proxy, overwrite or append `X-Forwarded-For` and set `TRUSTED_PROXY_HOPS` to match; apply the body size limits.
- Allow inbound traffic to ports 80 and 443 only. Never publish the database port.
- Restrict outbound traffic from the server to the services the instance uses (SMTP, LDAP, ACME, Docker Hub and npm during builds).

14.2.2 Secrets

- `.env` has mode 600 and is owned by the account that runs Docker.
- `AUTH_SECRET`, `POSTGRES_PASSWORD`, `APP_DB_PASSWORD` and the optional keys were generated with `openssl rand`, are unique to this instance, and are stored in the password vault.
- `INITIAL_ADMIN_PASSWORD` is removed from `.env` and from the `migrate` container.
- Shell access to the server, membership of the `docker` group and the database owner's password are limited to the people who operate the instance. Any of them can bypass every control in the application, including the audit log's protections.

14.2.3 Rotating AUTH_SECRET

Rotate `AUTH_SECRET` if it may have been exposed.

Warning: Rotating `AUTH_SECRET` ends every session at once. Without `SETTINGS_ENCRYPTION_KEY` and `BACKUP_SIGNING_KEY` it also discards the stored integration secrets and the ability to restore older in-app snapshots. Read the table before you rotate.

The consequences:

Effect	Without the dedicated key	With the dedicated key set
Every session ends; everyone signs in again.	Always	Always
SAML sign-ins in progress fail and must be started again.	Always	Always
Stored SMTP password, LDAP bind password and SAML private key become unreadable and must be entered again under Settings → Integrations.	Yes	No, if <code>SETTINGS_ENCRYPTION_KEY</code> is set
In-app snapshots taken before the change can no longer be restored in the application.	Yes	No, if <code>BACKUP_SIGNING_KEY</code> is set

Server backups made with `deploy/backup.sh` are unaffected. To rotate:

1. Generate the new value: `openssl rand -base64 32`.
2. Replace `AUTH_SECRET` in `.env` and in the vault.
3. Apply it: `docker compose --profile tls up -d`.
4. Sign in and re-enter the integration secrets if needed; send a test email.

`SETTINGS_ENCRYPTION_KEY` and `BACKUP_SIGNING_KEY` are rotated the same way, with the effects in their own rows above.

14.2.4 Backups

- `deploy/backup.sh` runs nightly and its failures are alerted.
- Backups are copied off the server and encrypted with age, with the private key kept off the server.
- Access to the off-site copies is as restricted as access to the application's data.
- A restore drill has passed within the last quarter.

14.2.5 Accounts and access

- Every person has their own account; no shared accounts.
- Administrators are few. Everyone else gets rights through groups and roles at the lowest level that works (*own* before *all*, read before update). See Users, Groups and Roles.
- At least two break-glass administrators exist, with passwords in the vault, and their sign-ins are reviewed (each one emails a security alert to every administrator).
- Two-step verification is required under **Settings → Sessions** for password and LDAP sign-in, or **SSO only** is on if your identity provider enforces MFA.
- Session length and idle timeout under **Settings → Sessions** match your policy.
- API tokens have the narrowest scopes and a maximum lifetime set under **Settings → Sessions**; unused tokens are revoked. See API Access and Integrations.
- Local passwords, where used, follow your password policy. The product requires at least 8 characters for a local password, whether set by an administrator or changed by its owner on My profile.
- Separation of duties stays on unless you have a documented reason to relax it.

14.2.6 Monitoring

- SMTP works, so administrators receive security alerts.
- `/api/health` is monitored externally.
- The audit chain is verified daily and the chain head copied off the server.
- Audit log retention is set to your record-keeping requirement (365 days minimum). See Audit Log and Evidence Integrity.

14.2.7 Keeping up to date

- Upgrade to new releases promptly with `deploy/update.sh`; security fixes arrive only in releases.
- Keep the host operating system and Docker Engine patched.
- The `postgres:18-alpine` and `caddy:2.11-alpine` images are named by tag, so `docker compose pull` followed by `docker compose --profile tls up -d` picks up their patch

releases. If your change process requires pinned images, pin them by digest in `docker-compose.override.yml`:

```
docker pull caddy:2.11-alpine
docker image inspect --format '{{index .RepoDigests 0}}' caddy:2.11-alpine
```

Then set the service's `image:` to `caddy:2.11-alpine@sha256:...`. A pinned image never changes until you change the digest, so you then take on applying its security fixes yourself.

14.3 Known limits

These are properties of the design that the operator controls compensate for:

- Anyone with the database owner's password or root on the server can disable the audit triggers and rewrite entries along with their hashes. Copying the chain head off the server after each verification detects such a rewrite.
- The application role can add audit entries, so a compromised application process could add false entries, though it cannot change or remove existing ones.
- An unsigned snapshot (from before snapshot signing) can still be restored after an administrator confirms it; the restore is recorded as unsigned.
- Upload links for AI assistants carry their token in the URL path; they expire after 15 minutes.

Index

A

- Account linking 78
- Accounts and access 187
- Acknowledgement 56, 112
- Adding a local user 39
- AI assistants (MCP) 144
- API token 33, 57, 59, 83, 105, 141, 147, 149
- API tokens 141
- Appetite bands 101
- Approval workflow 50, 54, 103, 105
- Archive 165, 166
- Article 30 register 103
- Asset 47, 59, 93, 129, 137
- Attachment 184
- Attributes and account matching 74
- Audit chain verification fails 182
- Audit entries 69
- Audit finding 47, 93, 101, 121, 129
- Audit log 12, 52, 85, 149, 151, 154, 158, 177
- AUTH_SECRET 186
- Authentication 146

B

- Back up and count rows 172
- Background jobs 12
- Backup 160, 161, 162, 164, 165, 187
- Backups 187
- Before you begin 16
- Before you run it 135
- Before you start 73
- Before you upgrade 170
- Bootstrapping the first administrator 21
- Branding 90
- Build and start 172
- Built-in role matrices 61
- Bulk changes and rules 123
- Bundle format in brief 137
- Bundle loader 33, 101, 102, 123, 127, 134, 137
- Business unit 47, 59, 129, 137

C

- Caddy 9, 11, 15, 17, 18, 24, 180
- Caddy does not get a certificate 180
- Calculation matrix 100
- Changing the scale on a live register 101
- Changing your own password 65
- Choose how HTTPS is provided 18
- Choosing a method 127
- Classification 85, 100, 102, 106, 128, 129, 137
- Classifications 100
- Cleaning up after upgrades 174
- Comment 121, 123
- Common problems 179
- Compliance framework packages 132
- Components 9
- Configuring SMTP 108
- Container hardening 13
- Container health checks 176
- Control 52, 59, 85, 93, 95, 101, 112, 129
- Control testing 52, 101, 112, 154
- Control testing settings 101
- Copying backups off the server 162
- Create the .env file 17
- Creating a custom role 51
- Creating a rule 121
- Creating a token 141
- Creating and changing fields 97
- Creating and editing a group 44
- CSV import 97, 127, 128, 131, 132, 137, 149
- CSV imports 128
- Custom field 47, 50, 59, 85, 94, 95, 97, 98
- Custom fields 94

D

- Dashboard 21, 47, 56, 59, 60, 101
- Data Liaison 47, 50, 54, 56, 59, 61, 112
- Data processing agreement 112
- Database roles 12
- Deactivating and reactivating 41
- Delegating user administration 43
- Deleting a group 45
- Deleting a user 42
- Deliverability 125
- Demo data 138
- Development and demonstration only 34
- Disk full 181
- Docker Compose 9, 14, 125, 172, 173, 180, 181, 187
- DPIA 47, 50, 54, 103, 105, 112

E

- Editing a template 117
- Editing a user 40
- Effect of changes on open sessions 57
- Email is not sent 181
- Encrypting backups 161
- Endpoint and tokens 80
- Enrollment 67
- Environment variable 17, 19, 24, 26, 27, 167
- Evidence 47, 59, 112, 145, 149, 158
- Evidence file fingerprints 158
- Exactly how Own is evaluated 47
- Example: Microsoft Entra ID 76
- Example: Okta 77
- Exception 47, 50, 53, 54, 112, 118, 121, 129
- Expiry, revocation and restores 142
- Export 131, 153, 158, 165, 166
- Exporting 153
- Exporting and the scheduled snapshots 166

F

- Field types 97
- File rules 131
- Files moved onto evidence 158
- First sign-in 22
- First start 20
- First-time migration 137
- Framework content shipped with the source 133

G

- GDPR 133
- Group 36, 43, 44, 45, 47, 85
- Groups 43

H

- Hash chain 154, 172, 177, 182
- Health check 20, 140
- Health checks 176
- Host requirements 14
- How accounts come into existence 38
- How custom fields appear 98
- How email is sent 108
- How levels combine 47
- How rules are delivered 123
- How settings are stored and applied 85
- How the components connect 11
- How values reach the containers 26

I

- IdP-initiated sign-in 76
- Importing a package 132
- In-app snapshots 164
- Incident classifications 102
- Install the release 172
- Instance settings (.env) 27
- Integration points 140

K

Keeping an external record 159

Keeping up to date 187

Key rotation 68

Known limits 188

L

Label 85, 93

Labels 93

LDAP 69, 15, 40, 64, 78, 149, 184

Limits 82

Limits, auditing and reachability 147

Local passwords 64

Locked out of administrator accounts 182

Logs 177

M

Magnitudes 100

Map of the settings pages 85

Mapping requirements 134

MCP 85, 140, 144, 145, 156

migrate fails 179

Migration 9, 11, 12, 20, 27, 137, 173, 179

Modules 59

Monitoring 187

N

Network ports 15

NIST SP 800-53 132, 133, 138, 158

Notification rule 12, 35, 59, 85, 108, 111, 121, 138

Notification rules 121

Notification template 47, 111

O

Operator checklist 185

Operator script variables 33

Option 1: the bundled Caddy 18

Option 2: the bundled Caddy with your own certificate 18

Option 3: your own reverse proxy 19

Owner 42, 44, 47, 52, 129, 131, 137, 154

P

Password 27, 42, 64, 65, 78, 108, 149

Password policy 64

Permission reference 58

Permissions 36, 46, 58, 143, 149

Personal data breach 123, 132

Plan the file changes 171

Policy portal 50, 56, 59

Policy settings 102

Post-installation checklist 24

Privacy settings 103

Q

Questionnaire 47, 59, 112, 128

R

- Rate limit 27, 82, 143, 184
- Recalculate all risks 101
- Record of processing 47, 95, 112
- Record types and columns 129
- Remediation project 9, 24, 27, 47, 59, 93, 95, 129
- Report schedule 108, 123, 124, 138
- Requiring two-step verification 68
- Resetting a password 42
- Resetting a user's two-step verification 68
- Residual risk 137
- Restore 33, 142, 160, 162, 163, 166, 167, 173
- Restore drill 167
- Restoring a server backup 162
- Restoring a snapshot 166
- Restoring in place on the same server 163
- Retention 156
- Review 47, 50, 54, 112, 118, 127
- Rights on configuration modules 60
- Risk appetite 100, 101, 112
- Risk score 100, 129
- Risk Settings 98
- Roles 46
- Rollback 170, 172, 173, 174
- Rolling back 173
- Rotating AUTH_SECRET 186
- Rotating the IdP certificate 77
- Row counts 178
- Running an update 170
- Running it 135
- Running it again 136
- Running several instances on one server 24

S

- SAML 64, 72, 74, 76, 78, 149, 181
- SAML single sign-on 72
- Scheduled reports delivery 123
- Scheduler 118, 12, 112, 124, 187
- Scheduling 161
- SCIM 40, 45, 78, 79, 80, 82, 149
- SCIM provisioning 79
- SCIM-managed groups 45
- Searching and filtering 152
- Secrets 186
- Security alerts 156
- Separation of duties 54, 105
- Server backups 160
- Service account 69, 147
- Service accounts and token practice 147
- Session 40, 57, 65, 83, 85, 105, 184, 187
- Sessions and sign-out 83
- Sessions page: token lifetime, storage and separation of duties 105
- Set by Compose and the image 32
- Setting up Microsoft Entra ID provisioning 82
- Settings 74
- Settings checklist for a new instance 106
- Settings that live in the application 35
- Sign-in methods at a glance 64
- Sign-in throttling and lockout 66
- Signing 165
- Single sign-on errors 181
- SMTP 16, 108, 110, 124, 125, 140, 156, 181
- SSO only and break-glass administrators 78
- Staff and portal-only users 56
- Storage 65
- Supported browsers 14
- Supported operations 81

T

- Tag a rollback image 172
- Template keys and their variables 112
- Templates 111
- Testing email 110
- The Approve right 50
- The audit chain 177
- The Audit Log page 151
- The built-in roles 50
- The bundle loader 134
- The delivery log 124
- The hash chain 154
- The health endpoint 176
- The Notifications page 111
- The permission model 46
- The REST API for administrators 143
- The Review right 50
- The sign-in sequence 65
- The sign-in step 67
- The stack does not start: a variable is not set 179
- Third party 47, 59, 93, 95, 100, 129, 137, 145
- Throttling 66, 67, 149
- Time zone 74, 82, 118
- TLS and the network 185
- Transfer impact assessment 112
- Treatment 47, 135, 137
- Troubleshooting 125
- Turning a template off 118
- Turning it on 145
- Two-step verification 67, 65, 68, 69, 149

U

- Unpack the release 16
- Upgrade 14, 16, 18, 50, 161, 170, 174
- Upgrading an installation from before October 2026 174
- Upload link 147
- User 36, 38, 39, 40, 42, 43, 56, 68
- User accounts 38
- Using the audit log as evidence 158

V

Verify and record 172
Verifying the chain 155
Version 14, 132, 170, 172
Viewing a change 153

W

web is unhealthy or keeps restarting 180
What a group does 44
What a new installation already contains 128
What a new installation contains 100
What a response must satisfy 75
What a snapshot contains 164
What a token can do 142
What an assistant can do 145
What an import does 132
What else to keep 162
What else to watch 177
What is logged 149
What provisioning may and may not do 82
What the product enforces 184
What the update does 171
What to do if verification fails 155
When an update fails 173
Where backups are assembled 165
Where custom fields apply 95
Where identity is administered 36
Worked example: auditor, read-only 52
Worked example: control owner 52
Worked example: delegating exception decisions 53
Worker 9, 11, 12, 108, 110, 118, 125, 181