FOCUS FAMILY

Technical Handbook

Operating, securing and administering the server

Version 1.0.0 · September 2026 · Software Notion

Contents

About this handbook

This handbook is written for **IT administration and technical service providers** who operate an organisation's Focus Family server. It covers architecture, installation, configuration, hardening, backup, upgrades and troubleshooting – plus the technical areas of the admin panel.

Using the panel for day-to-day administration is described in the **Admin Panel Handbook**; using the apps in the two app handbooks.

Assumed knowledge: Linux basics, Docker and Docker Compose, a reverse proxy (nginx, Caddy or Traefik) and handling TLS certificates.

Principles of the operating model

- **One organisation runs one server instance** in its own infrastructure. There is no vendor cloud.
- The stack needs **no connection** to any analytics, crash or AI service of the vendor.
- The only optional external contact is the **licensing server** for the organisation licence.
- The apps work **entirely offline**; the server exists for teamwork and administration.

Architecture

The stack consists of four containers:

Service	Purpose
db	PostgreSQL 17, not publicly reachable, named volume `pgdata`
migrator	one-shot at start-up: creates the schema or applies pending schema migrations and provisions the app role
clamav	malware scanning for all uploads, named volume `clamdb` for the signatures
app	the server itself, bound to `127.0.0.1:8080` behind the organisation's reverse proxy

Data and volumes

Volume	Content	Consequence if lost
`pgdata`	PostgreSQL data	complete data loss
`blobs`	attachments, encrypted at rest	documents and media lost
`dpkeys`	ASP.NET data protection keyring	all sessions and antiforgery tokens invalid
`clamdb`	ClamAV signatures	re-downloaded (uncritical)
`licstate`	identity of the licensing installation	re-provisioning with the licensing operator required
`deploy/docker/keys`	signing and encryption certificates (read-only)	sign-in no longer possible

> **Rule of thumb:** `pgdata`, `dpkeys` and `licstate` together form the **identity of the installation**. Deleting them turns a restore into a re-provisioning.

Database roles

- `fokus_migrator` owns the schema and may run DDL. Only the migrator container uses this role.
- `fokus_app` runs the application with **DML rights only**; `CREATE TABLE` and `DROP` are denied.

The application itself **never** executes DDL. Schema changes only happen in the controlled migrator step.

Requirements

- **Server:** Linux (amd64/x86_64) with Docker Engine and Compose v2.
- **Recommendation:** 4 CPU cores, 8 GB RAM and 50 GB storage as a starting point. ClamAV alone needs roughly 1.5–2 GB RAM.
- **Network:** a domain pointing to the server and a TLS certificate (institutional PKI or Let's Encrypt) at the reverse proxy.
- **Certificates:** production **signing and encryption certificates** in PKCS#12 format for token issuance.
- **Licence:** an **enrolment key** from the licensing operator if an organisation licence is used.

> In a development environment the server starts with ephemeral keys. In production the certificates are **mandatory** – without them the application deliberately refuses to start.

Installation step by step

1. Copy the files to the server

Copy the deployment directory (compose file, `.env.example`, `keys/`, proxy examples, backup scripts) to the server, for example to `/opt/focusfamily`.

The server image is built **once on a build machine** and shipped as a tarball – no .NET SDK is needed on the server:

```
docker load -i image/fokusfamilie-server-1.0.0-amd64.tar.gz
```

2. Create the configuration

```
cp .env.example .env
uuidgen                     # use the value as SERVER_INSTANCE_ID
nano .env                   # replace every CHANGE_ME
```

3. Create certificates

```
mkdir -p deploy/docker/keys
openssl req -x509 -newkey rsa:2048 -nodes -keyout /tmp/k.pem -out /tmp/c.pem \
  -days 825 -subj "/CN=fokusfamilie"
openssl pkcs12 -export -inkey /tmp/k.pem -in /tmp/c.pem \
  -out deploy/docker/keys/signing.pfx    -passout pass:"$CERT_PASSWORD"
openssl pkcs12 -export -inkey /tmp/k.pem -in /tmp/c.pem \
  -out deploy/docker/keys/encryption.pfx -passout pass:"$CERT_PASSWORD"
```

For real deployments use your institutional PKI instead of a self-signed certificate.

4. Signing key for offline grants

```
{ printf '\nOFFLINE_GRANT_KEY="'; openssl ecparam -genkey -name prime256v1 -noout; printf
'"\n'; } >> .env
```

If the value stays empty, the server generates an ephemeral key at every start and logs a warning – offline grants then do not survive a restart.

5. First start with bootstrap

For the very first start, enable the one-time bootstrap in `.env`:

```
BOOTSTRAP_ENABLED=true
BOOTSTRAP_ADMIN_EMAIL=admin@your-organisation.org
BOOTSTRAP_ADMIN_PASSWORD=<one-time password>
```

```
docker compose up -d
docker compose logs -f migrator            # watch the schema being created
curl http://127.0.0.1:8080/health/ready    # expected: 200
```

The migrator creates the schema, then the application creates the first organisation admin with a forced password change. The initial password is **never** logged.

As soon as signing in to the panel works:

```
BOOTSTRAP_ENABLED=false
```

```
docker compose up -d
```

6. Set up the reverse proxy

Terminate TLS at the proxy and forward to `app:8080`. Examples for nginx and Caddy ship with the deployment directory.

Important:

- forward `X-Forwarded-Proto`, `X-Forwarded-Host` and `X-Forwarded-For`,
- set `REVERSE_PROXY_IP` in `.env` to the proxy address – forwarded headers are only accepted from there,
- choose a `client_max_body_size` (nginx) large enough for attachments.

Configuration reference

The compose file reads the following variables from `.env`. Never put `.env` under version control; values may also come from Docker secrets or a secret store.

Variable	Purpose
`POSTGRES_PASSWORD`	password of the DDL role `fokus_migrator`
`FOKUS_APP_PASSWORD`	password of the least-privilege role `fokus_app`
`SERVER_INSTANCE_ID`	stable GUID identity of this server instance
`ORGANIZATION_NAME`	display name of the organisation
`BOOTSTRAP_ENABLED`, `BOOTSTRAP_ADMIN_EMAIL`, `BOOTSTRAP_ADMIN_PASSWORD`	one-time first-admin bootstrap
`BOOTSTRAP_FACHADMIN_EMAIL`, `BOOTSTRAP_FACHADMIN_PASSWORD`	optional bootstrap of a professional admin account
`SERVICE_CLIENT_ID`, `SERVICE_CLIENT_SECRET`	optional confidential service client
`MOBILE_REDIRECT_URI`	additional OAuth redirect URI for the mobile client
`CERT_PASSWORD`	password of the PKCS#12 certificates
`REVERSE_PROXY_IP`	the only address whose `X-Forwarded-*` headers are trusted
`BLOB_ENCRYPTION_KEY_BASE64`	**mandatory:** AES-256 key (32 bytes base64) for attachments and data copies
`OFFLINE_GRANT_KEY`	private EC P-256 key (PEM) for signing offline grants
`OFFLINE_GRANT_KEYSET_VERSION`	monotonic version of the offline grant keyset (default 1)
`OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY`	additional trusted public key during a rotation
`LICENSING_INSTALLATION_KEY`	one-time enrolment key from the licensing operator
`FileScanning__Enabled`, `FileScanning__Host`	configure malware scanning (see below)

Blob encryption: generate the key with `openssl rand -base64 32` and store it **separately from the blob archive**. Without it the archive is unreadable. Development environments may set `BlobStorage:AllowGeneratedKey=true` – never in production.

Malware scanning: `FileScanning__Enabled=true` plus `FileScanning__Host` enables ClamAV. If **nothing** is configured in production, the server behaves **fail-closed**: uploads are rejected rather than accepted unscanned.

Database and migrations

There are **no EF migration folders**. A hand-written `MigrationRunner` does the work instead:

- **Fresh database:** `EnsureCreated` builds the complete schema.
- **Existing database:** pending versions from the `SchemaMigrations` ledger are applied in **one transaction** under `pg_advisory_xact_lock`; the version stamp is written in the same transaction. Competing migrators serialise themselves that way.
- Afterwards the migrator provisions the `fokus_app` role and exits. The application only starts after the migrator succeeded.

The migrator runs on **every** `docker compose up` – no extra manual step is needed.

Backup and restore

> A backup only counts as working once its **restore has been tested**. And a database backup **without the cryptographic keys** cannot be fully restored.

What has to be backed up

No.	Item	Frequency
1	**PostgreSQL** – all content, users, teams, assignments, change feed, audit, policies	periodic
2	**Blob storage** – attachments from the `blobs` volume	periodic
3	**Signing/encryption certificates** plus `CERT_PASSWORD`	rarely, in the secret manager
4	**Offline grant key** plus `OFFLINE_GRANT_KEYSET_VERSION`	rarely, in the secret manager
5	**Data protection keyring** (`dpkeys`)	rarely
6	**Licensing installation state** (`licstate`)	rarely

Item 2 is unreadable without the **blob key** (`BLOB_ENCRYPTION_KEY_BASE64`) – store both **separately**.

Backup

```
./deploy/docker/backup.sh      # writes ./backups/db-<stamp>.dump + blobs-<stamp>.tgz
```

`db-*.dump` is a PostgreSQL dump in custom format, streamed straight out of the `db` container. Push the results onward encrypted and versioned (restic, borg, object storage). The product contains **no** static backup password.

Restore

Into a fresh stack (remove the old one with `docker compose down -v` first):

```
DB_DUMP=./backups/db-<stamp>.dump BLOBS=./backups/blobs-<stamp>.tgz ./deploy/docker/restore.sh
curl http://127.0.0.1:8080/health/ready     # expected: 200
```

The script starts an empty database, restores the dump, restores the blobs, runs the migrator (applying pending schema steps and re-provisioning the `fokus_app` role – roles are cluster-wide and not part of a dump) and starts the application.

Test restore

1. Create sample data (for example a synced case).
2. `./deploy/docker/backup.sh`.
3. `docker compose down -v`.
4. Run `restore.sh` with dump and blobs.
5. Restore keys, certificates and `licstate` from the secret manager.
6. Verify: health ready, sign-in works, the sample case is present, attachments open, and the checksums of finalised documents still match.

Version compatibility

Restore into the **same or a newer** server version. A newer server applies pending schema steps at start-up. **Never** restore a newer dump into an older server.

Upgrades

Upgrades are **backup-first** and never destructive.

```
./deploy/docker/backup.sh                 # 1. back up
docker compose pull                       # 2. pull the new image
docker compose up -d                      # 3. migrator, then the app restarts
curl http://127.0.0.1:8080/health/ready   # 4. smoke test
```

Notes

- **Schema changes** only happen in the migrator step, never under the app role. Check the release notes before a schema-changing upgrade.
- **API compatibility:** server N supports clients N and N-1 as far as the schema allows. The policy **minimum client version** lets you lock out older clients softly or hard. Local data always remains accessible to the clients.

- **Rolling upgrade:** no critical session state is kept in memory – keys, keyring and blobs live on shared volumes. Several app instances behind the proxy can be replaced one after another; the migration transaction serialises via the advisory lock. Version 1 ships as a single instance.
- **Rollback:** restore the pre-upgrade backup into the previous image version.

Licensing

If your organisation uses an organisation licence, the installation registers with the **licensing server**. This happens **lazily**: nothing happens at start-up; only at the first licence contact does the server generate its EC P-256 key pair, redeem the **single-use** `LICENSING_INSTALLATION_KEY` and persist the resulting installation identity in the `licstate` volume.

> **Never** delete the `licstate` volume on an already enrolled server – the enrolment key cannot be redeemed twice. When moving a deployment, take `licstate` with you.

Alternatively you can provision manually: `--licensing-keygen` prints a fresh installation ID and a key pair; hand the public key to the licensing operator.

Licence seats are assigned exclusively by administration in the panel. Devices never claim a seat automatically. Without a seat a device cannot sync.

Security

Transport and hardening

- Production is **HTTPS only**; TLS terminates at the proxy.
- **Forwarded headers** are accepted only from the configured `REVERSE_PROXY_IP`. This keeps the per-client-IP **rate limiting** from being bypassed with forged headers.
- **Security headers, CSP, HSTS** and **antiforgery** in the panel's cookie area are active.
- **Rate limiting** per IP with fixed windows, stricter for sign-in and device enrolment.
- Signing and encryption certificates are **mandatory** (fail-fast at start-up if missing).
- In production, sensitive EF logging and the developer exception page are disabled; diagnostics never reveal connection strings and logs never contain tokens, secrets or case content.

Authorisation

Every read and write is authorised **on the server** against the current state, never against client claims. Visibility is already filtered in SQL. Entries marked **private** are invisible to other app users – including team leads and all admin roles – in queries, search, export and direct access by ID.

Device binding

Devices register through a challenge and proof-of-possession flow. Every API request from a bound device carries a signed **device proof** whose nonce is persisted (replay protection); access tokens are bound to the device. For reinstalls, administration issues **re-enrolment codes**.

Authentication

- **OpenIddict** with authorization code + PKCE, short-lived and rotated tokens.
- **MFA** via TOTP as a global policy with a challenge at sign-in – for apps and panel.
- **Passkeys** (WebAuthn) as a passwordless alternative; while a passkey is active, password sign-in is disabled for that account.
- **Technical admin accounts** can never sign in to the apps.
- **Break-glass** is a time-limited (at most 120 minutes), MFA-enforced and audited exception session – and only possible if the organisation policy allows it.

Uploads

Uploads are limited by size and type and are scanned with ClamAV. In production the behaviour is **fail-closed**: if the scanner is not configured, unreachable or times out, uploads are rejected. Blob content is encrypted at rest.

Rotating the offline grant keyset

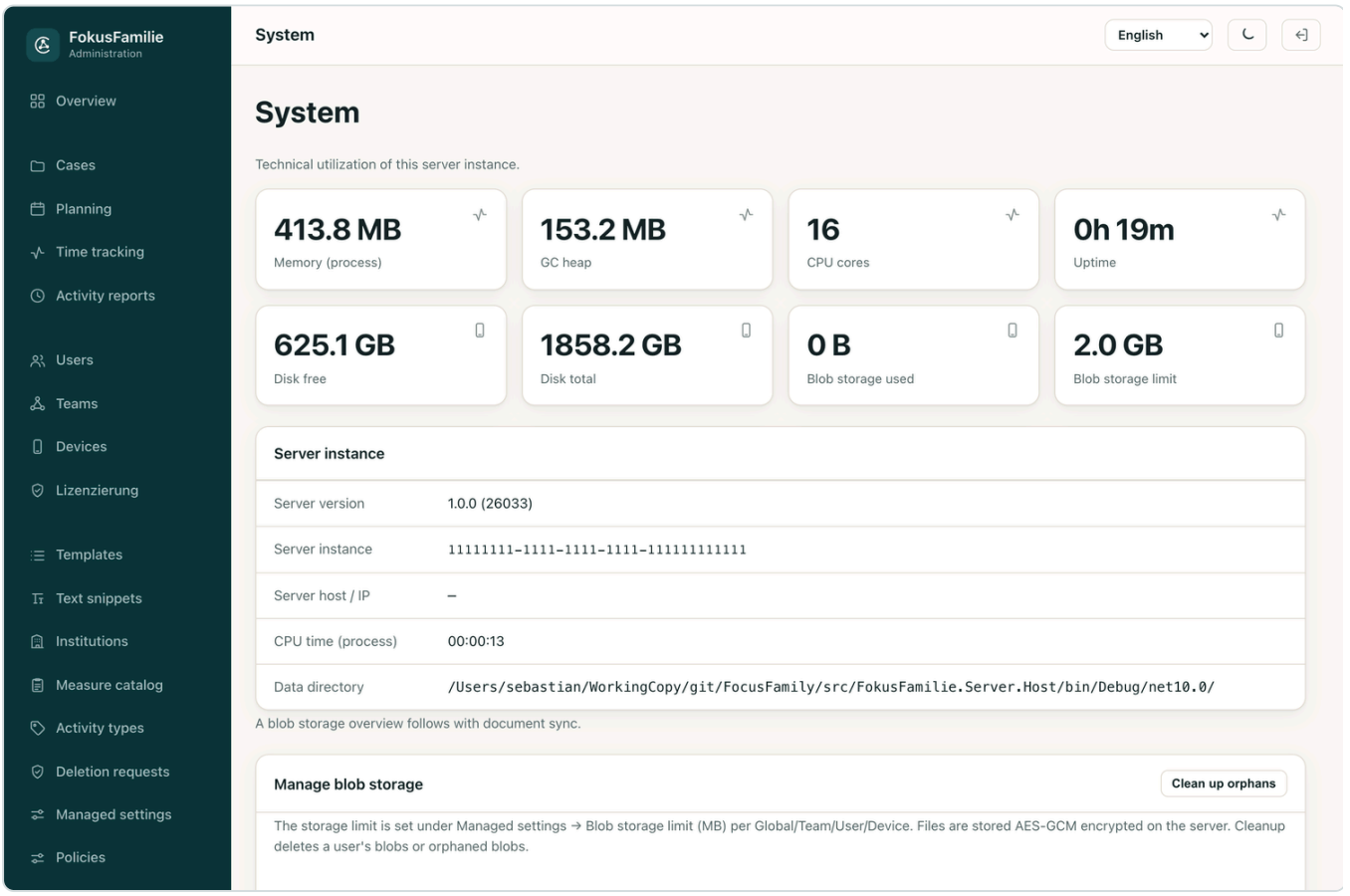
Clients only trust offline grant keys from a **signed keyset** bound to the `ServerInstanceId` and carrying a monotonic version. Rotation is a **two-phase** operation:

1. Keep the old private key as the signer, add the **new public** key to `OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY` and raise `OFFLINE_GRANT_KEYSET_VERSION`. Wait until all clients have synced at least once.
2. Switch `OFFLINE_GRANT_KEY` to the new private key, keep the **old public** key for the transition and raise the version again. Remove the old key (and raise the version once more) once the longest offline grant lifetime has passed.

> If the version is **not** raised when the key list changes, clients keep their existing keyset (fail-closed). Never lower the version.

Technical areas of the admin panel

System status

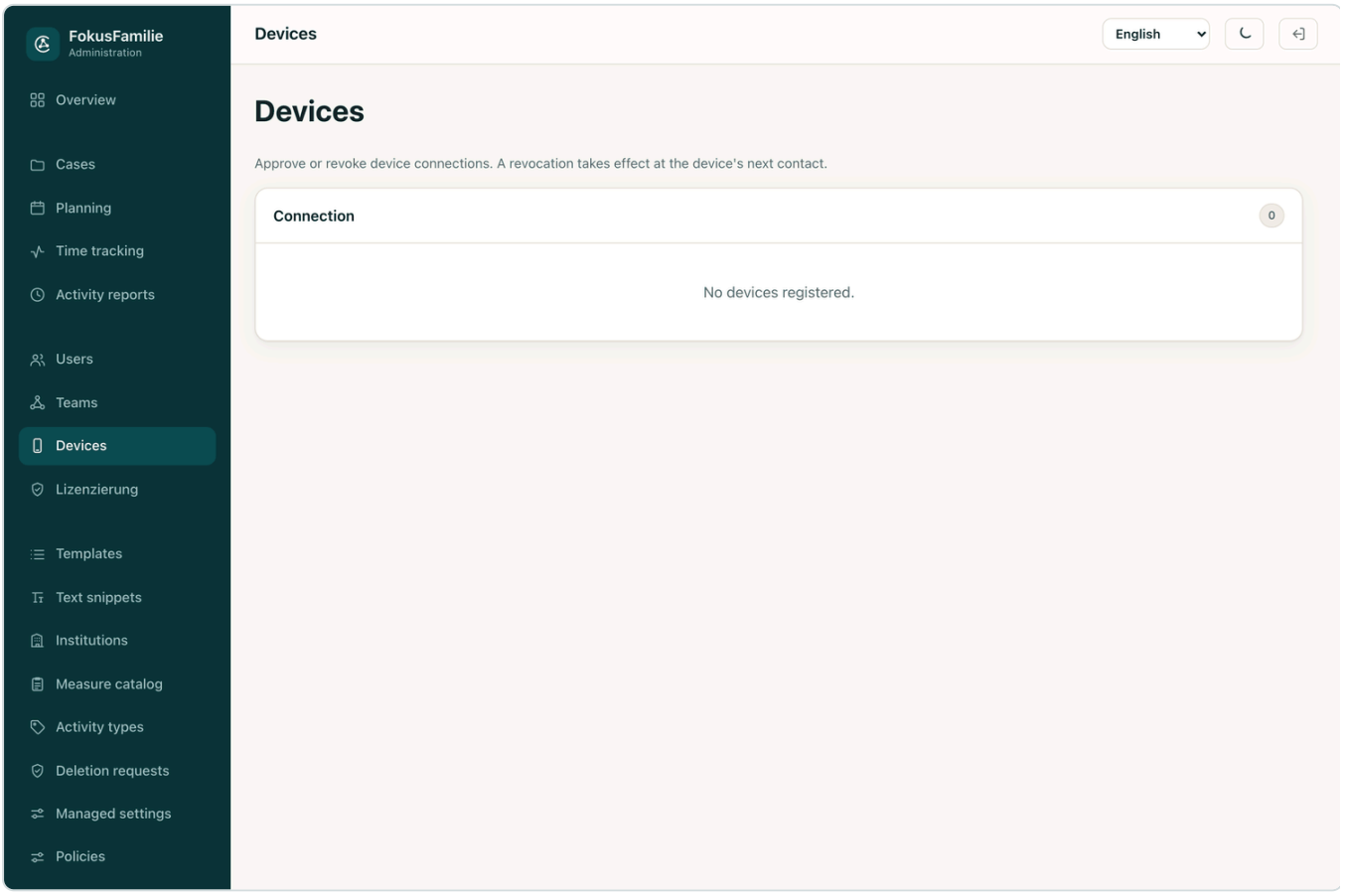


Server version, instance, storage usage and the blob storage administration.

The menu item **System** (permission *View system status*) shows the server version, API version, organisation, server instance ID, change sequence and load. It also hosts the **blob storage** administration.

> ⚠ **Cleaning deletes irreversibly.** The function removes a user's uploaded documents and media – or orphaned files – **permanently** from the server. Content that no longer exists on any device is then lost. A confirmation prompt appears first.

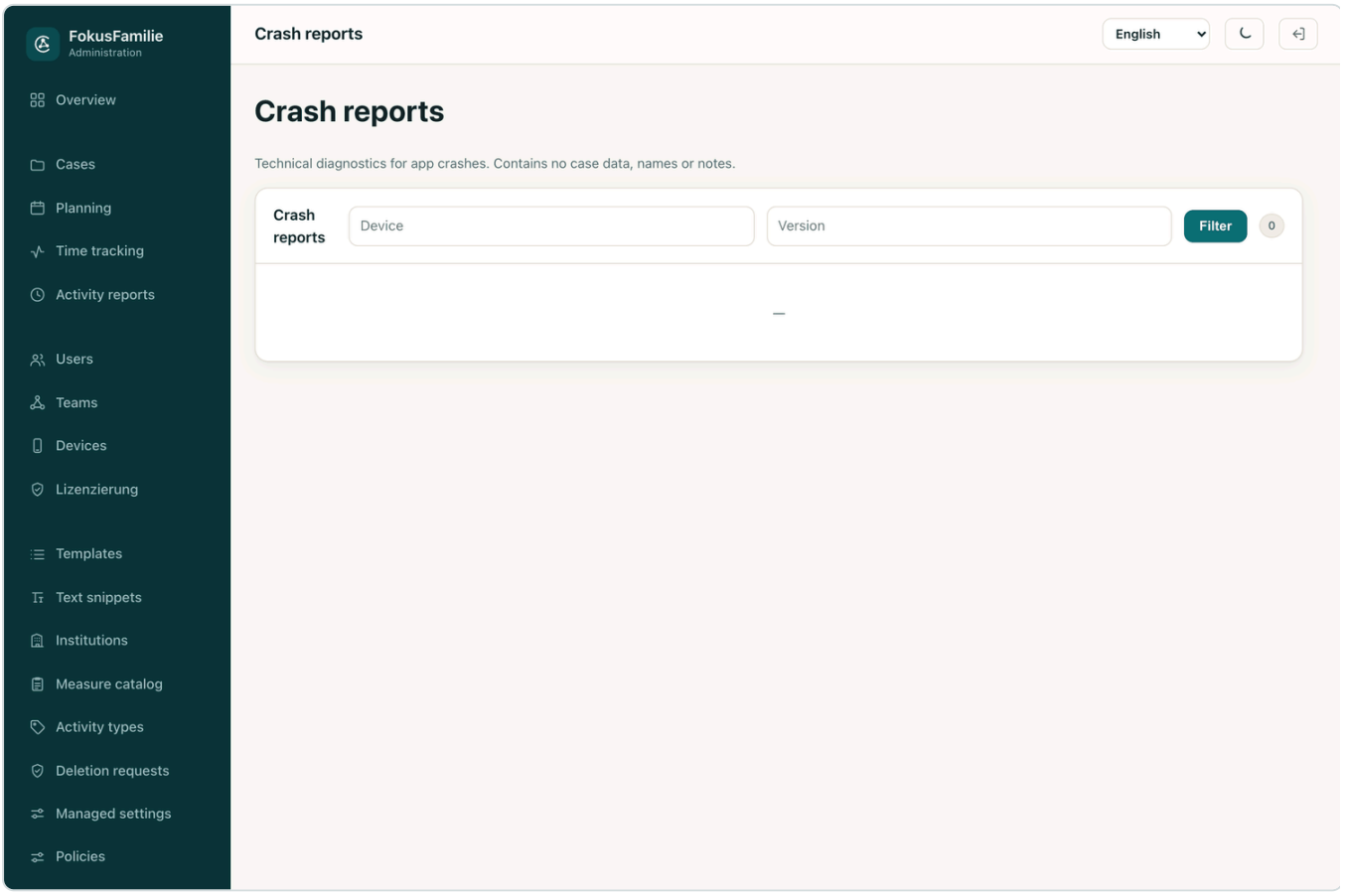
Devices



Approve, block, reset the PIN lock, remove a device and issue re-enrolment codes.

Blocks take effect the next time the device contacts the server. **Remove** dissolves the binding, revokes the device registration of the licence and frees the seat.

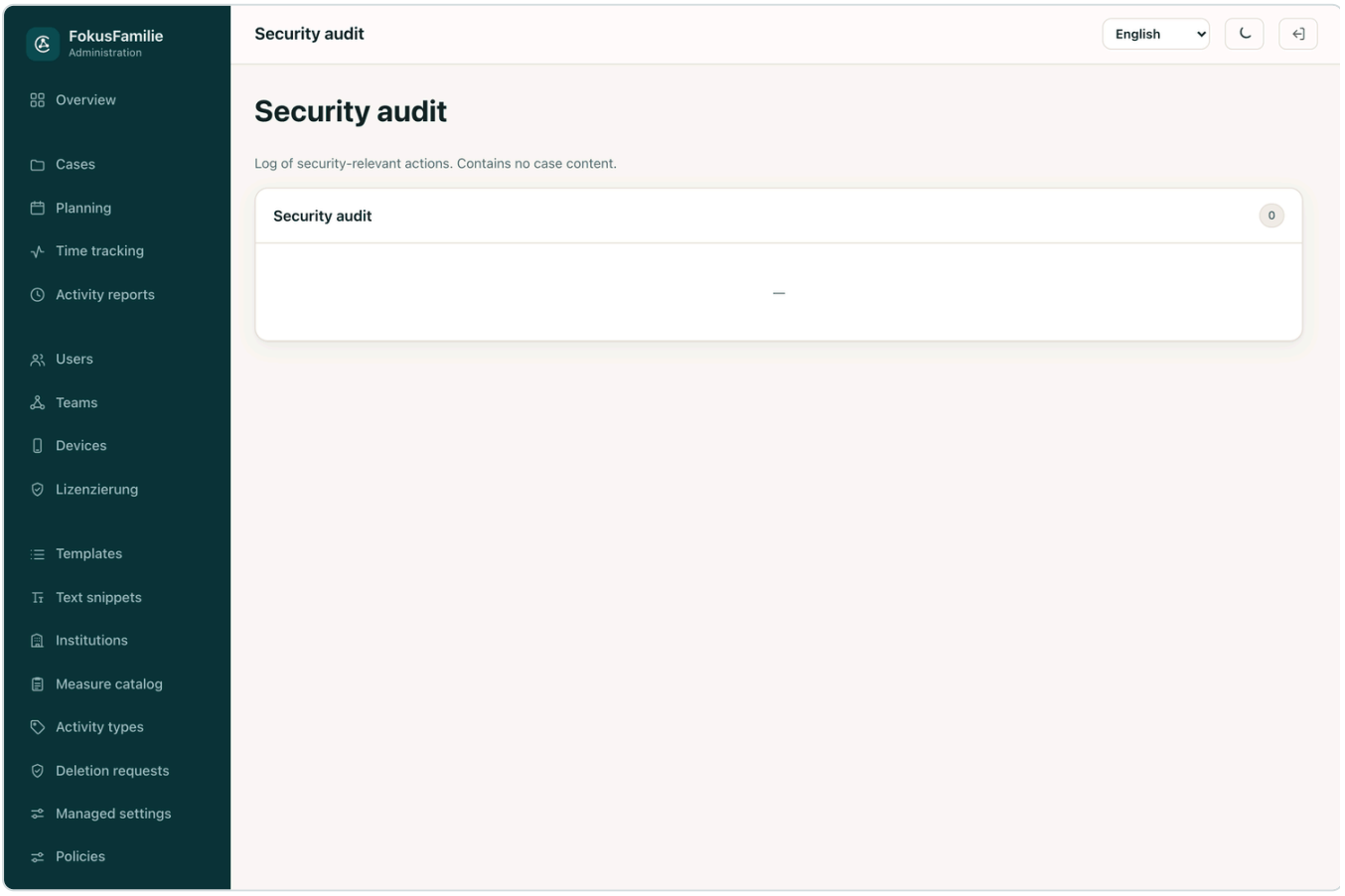
Crash reports



Technical error reports from the apps – without case data, names or notes.

Apps can send a technical report after a crash (permission *View crash reports*). The reports contain time, platform and technical details – **no** content.

Security audit



Who did what and when – including the result (success, denied, error).

The audit is your first stop for suspicious activity: sign-ins, permission changes, device approvals, policy changes, emergency access, storage cleaning and management corrections to time entries. It contains no case content.

Licensing

The menu item **Licensing** shows the state of the organisation licence, used and free seats and the registered devices. If licensing is not configured, the page reports that `Licensing:InstallationId` and `Licensing:PrivateKeyPem` are missing (generate a key pair with `--licensing-keygen`).

Operations and monitoring

Health endpoints

```
curl http://127.0.0.1:8080/health/live     # process is alive
curl http://127.0.0.1:8080/health/ready    # process plus database ready
```

Wire `/health/ready` into your monitoring. The container also ships its own `HEALTHCHECK`.

Logs

```
docker compose logs -f app
docker compose logs migrator
docker compose logs clamav
```

Logs **never** contain tokens, secrets or case content.

Monitoring

Monitoring is the organisation's responsibility (Prometheus, Grafana, Loki or OpenTelemetry, for example) and by default sends nothing to external providers. Useful signals: availability of `/health/ready`, response times, error rates, disk usage of `pgdata` and `blobs`, the age of the last backup and the ClamAV state.

Incidents

Lost or stolen device

1. Under **Devices** in the panel, **remove** the device – the binding is dissolved and the licence seat freed.
2. If compromise is suspected, also **deactivate the user account** or reset the password, and reset MFA or passkeys.
3. Trace the incident in the audit.

The data on the device stays protected by the app lock and local encryption.

Practitioner locked out of the app lock

Devices → **Reset PIN lock**. The lock is released at the next sync and the person sets a new PIN.

Compromised user account

Deactivate the account, reset the password, reset MFA and passkeys, review the devices and remove them if needed. Every step appears in the audit.

Compromised server keys

Replace the signing and encryption certificates and restart the application; all tokens become invalid and clients sign in again. Change the offline grant key using the described **two-phase rotation**.

Database or backup theft

Blob content is encrypted at rest; without the blob key attachments stay unreadable. Rotate database passwords and use the audit to see which accounts might be affected. Document the incident for GDPR notification duties.

Suspected malicious upload

ClamAV already refuses affected files during upload. Check the scanner logs and inform the practitioner concerned.

Troubleshooting

Symptom	Cause and remedy
`/health/ready` returns 503	database unreachable or migrator did not run. Check `docker compose logs migrator`; do `POSTGRES_PASSWORD` and `FOKUS_APP_PASSWORD` match?
App exits at start (production)	`deploy/docker/keys/*.pfx` or `CERT_PASSWORD` missing
401 or redirect loops behind the proxy	`X-Forwarded-Proto=https` is not forwarded, or `REVERSE_PROXY_IP` does not cover the proxy
Licensing reports the enrolment key as already used	the `licstate` volume was deleted after enrolment – restore it from backup or request a new key
Uploads stay in quarantine	ClamAV is not ready yet (the first signature download takes minutes). In production scanning is deliberately fail-closed
Devices do not sync	device not approved, licence seat missing, or the minimum client version locks the app out
Panel reports "licensing not configured"	set `Licensing:InstallationId` and `Licensing:PrivateKeyPem`
Build fails with `**/*.resx` (Docker Desktop for macOS)	known MSBuild quirk under virtiofs – build on a Linux host or use the prepared image

---

Checklists

Commissioning

- [ ] domain, DNS and TLS certificate in place
- [ ] image loaded, `.env` complete, no `CHANGE_ME` values left
- [ ] signing and encryption certificates created, `CERT_PASSWORD` set
- [ ] `BLOB_ENCRYPTION_KEY_BASE64` generated and stored **separately**
- [ ] `OFFLINE_GRANT_KEY` persisted
- [ ] `REVERSE_PROXY_IP` set, proxy forwards the forwarded headers
- [ ] first admin created via bootstrap, then `BOOTSTRAP_ENABLED=false`
- [ ] `/health/ready` returns 200, sign-in to the panel works
- [ ] licence enrolment done, `licstate` backed up
- [ ] backup script scheduled and a **test restore** performed
- [ ] monitoring for `/health/ready` and disk usage in place

Monthly routine

- [ ] backups exist and are no older than agreed
- [ ] test restore into a staging environment
- [ ] check server version and security updates
- [ ] review the audit for anomalies

- [ ] tidy up the device and seat lists
- [ ] check disk usage of `pgdata` and `blobs`

Appendix

Important paths and ports

Item	Value
App in the container	port 8080, bound to `127.0.0.1`
PostgreSQL	internal compose network only
Blob storage	`/var/lib/fokusfamilie/blobs` (volume `blobs`)
Data protection keys	`/home/app/.aspnet/DataProtection-Keys` (volume `dpkeys`)
Licensing identity	`/var/lib/fokusfamilie/licensing/installation.json` (volume `licstate`)
Certificates	`deploy/docker/keys/*.pfx` (mounted read-only)

Useful commands

```
docker compose ps                       # state of all services
docker compose logs -f app              # application log
docker compose restart app              # restart the application only
docker compose down                     # stop the stack (volumes stay)
docker compose down -v                  # remove the stack including volumes (data loss!)
./deploy/docker/backup.sh               # create a backup
```

Further documentation in the repository

`docs/server/DEPLOYMENT.md`, `docs/server/DATABASE.md`, `docs/server/BACKUP-RESTORE.md`, `docs/server/UPGRADE.md`, `docs/server/SECURITY.md`, `docs/server/AUTHENTICATION.md`, `docs/server/AUTHORIZATION.md`, `docs/server/SYNC-PROTOCOL.md`, `docs/server/INCIDENT-RESPONSE.md`, `docs/server/THREAT-MODEL.md` and `deploy/docker/README.md`.

Focus Family · Technical Handbook · Version 1.0.0 · September 2026