FOCUS FAMILY

Technical Handbook

Operating, securing and administering the server

Version 1.4.0 · November 2026 · Software Notion

Contents

1. **About this handbook**
    1. Principles of the operating model
2. **Architecture**
    1. Data and volumes
    2. Database roles
3. **Requirements**
4. **Installation step by step**
    1. 1. Copy the files to the server
    2. 2. Create the configuration
    3. 3. Create certificates
    4. 4. Signing key for offline grants
    5. Internal certificate authority (optional)
    6. 5. First start and initial administrators
    7. 6. Set up the reverse proxy
    8. Development machines only
5. **Configuration reference**
6. **Database and migrations**
7. **Backup and restore**
    1. What has to be backed up
    2. Backup
    3. Restore
    4. Test restore
    5. Version compatibility
8. **Upgrades**
    1. Updating from the admin panel
9. **Licensing**
10. **Single sign-on (SSO)**
    1. Prerequisites
    2. OpenID Connect or SAML 2.0?
    3. The "Single sign-on" page
    4. Microsoft Entra ID
    5. On-premises Active Directory via AD FS
    6. Keycloak
    7. Okta
    8. Google Workspace
    9. Other providers
    10. User accounts and linking
    11. Switching the sign-in method and removing the link

12. Emergency access and "Enforce SSO"
13. Sign-in re-confirmation
14. Intranet operation
15. Security and secrets
16. SSO troubleshooting
17. Audit events
18. SSO rollout checklist

11. Interfaces

12. Data room

13. Operation on the intranet or via VPN

1. Outbound connections of the server
2. Outbound proxy
3. Devices, licence and sync
4. DNS and certificates
5. VPN

14. Distributing the apps through device management

1. Android through Managed Google Play
2. iPhone and iPad through Apple Business Manager
3. Desktop app
4. Connecting to the server
5. Recommended device policies

15. Security

1. Transport and hardening
2. Authorisation
3. Device binding
4. Authentication
5. Uploads
6. Rotating the offline grant keyset

16. Technical areas of the admin panel

1. System status
2. Devices
3. Crash reports
4. Security audit
5. Activity record archive
6. Licensing

17. Operations and monitoring

1. Health endpoints
2. Logs
3. Checking time synchronisation
4. Monitoring

5. Sync retention and full resync

18. **Incidents**
    1. Lost or stolen device
    2. Practitioner locked out of the app lock
    3. Compromised user account
    4. Compromised server keys
    5. Database or backup theft
    6. Suspected malicious upload

19. **Troubleshooting**

20. **Checklists**
    1. Commissioning
    2. Monthly routine

21. **Appendix**
    1. Important paths and ports
    2. Useful commands
    3. Further documentation in the repository

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 and an optional fifth:

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
updater	optional (`COMPOSE_PROFILES=updater`): installs updates from the admin panel, the only service with access to the Docker socket

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.

In production (`ASPNETCORE_ENVIRONMENT=Production`) the server encrypts the keys in the `dpkeys` volume with the encryption certificate `encryption.pfx` (`CERT_PASSWORD`). On the first start after the update, the unencrypted keys written by version 1.0.0 are encrypted in place (log line "Encrypted … data-protection key file(s)"); sessions and protected data stay readable, and new keys are written encrypted right away. Reading the `dpkeys` disk alone is no longer enough to forge sessions. Backups taken before the update still contain the keys unencrypted and must be protected or deleted accordingly. The certificate therefore belongs to the installation identity: without `encryption.pfx`, sessions, the licensing installation identity and the SSO secrets can no longer be read. Back up `deploy/docker/keys/` together with `.env`.

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 DNS name for the server (public or internal only) and a TLS certificate (institutional PKI or Let's Encrypt) at the reverse proxy. The server does not have to be reachable from the internet; outbound it needs HTTPS to `fflicense.software-notion.com` (see "Operation on the intranet or via VPN").
- **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.

- **Time synchronisation:** the host must keep its clock synchronised continuously via NTP (chrony, systemd-timesyncd or ntpd, preferably against an internal time server). The containers have no clock of their own; they use the host's. Device proofs (±5 minutes), tokens, two-factor codes, SSO responses, licence checks, audit log timestamps and the retention jobs depend on it (see "Checking time synchronisation").

> 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/focusfamily-server-1.4.0-amd64.tar.gz
```

The loaded image is called `fokusfamilie-server`; this is a technical identifier referenced by the compose files and the update service.

2. Create the configuration

The partner package contains `setup.sh`, which generates `.env`, certificates and keys in one step (recommended). For a manual setup, copy the template and replace **every** `CHANGE_ME` value; each value is described in the template together with the command that generates it:

```
cp .env.example .env
chmod 600 .env
```

Value	What it is	Generate with
`POSTGRES_PASSWORD`	password of the PostgreSQL owner account `fokus_migrator`; only the migrator uses it (schema updates), nobody signs in with it	`openssl rand -hex 24`
`FOKUS_APP_PASSWORD`	password of the restricted runtime account `fokus_app` the panel uses for data access; the migrator sets it on every start	`openssl rand -hex 24`
`SERVER_INSTANCE_ID`	stable identity (GUID) of this installation; never change it after the first start	`uuidgen` (Linux without `uuidgen`: `cat /proc/sys/kernel/random/uuid`)
`ORGANIZATION_NAME`	display name of the organisation; in double quotes if it contains spaces	–
`BOOTSTRAP_ADMIN_EMAIL`, `BOOTSTRAP_ADMIN_PASSWORD` (plus `BOOTSTRAP_FACHADMIN_*`, `BOOTSTRAP_PRIVACYADMIN_*`, `BOOTSTRAP_ROOT_ADMIN`)	initial administrators, see step 5; the passwords are one-time passwords, any value is accepted	e.g. `openssl rand -base64 12`
`SERVICE_CLIENT_ID`, `SERVICE_CLIENT_SECRET`	optional confidential service client (client credentials); leave both empty if not needed	`openssl rand -hex 24`
`MOBILE_REDIRECT_URI`	additional sign-in redirect address of the apps; `https://<domain>/callback` with a domain, otherwise `http://localhost/callback`	–
`OPENIDDICT_ALLOW_INSECURE_HTTP`	`true` **only** for a test without TLS (panel at `http://127.0.0.1:8080`); always `false` behind HTTPS. With `false`, every cookie (session, antiforgery, language) gets the Secure attribute in production, so the admin panel only works over HTTPS	–
`CERT_PASSWORD`	password of the certificates `signing.pfx` and `encryption.pfx` (step 3)	`openssl rand -hex 16`
`BLOB_ENCRYPTION_KEY_BASE64`	AES-256 master key (32 bytes, base64) for attachments and privacy data copies	`openssl rand -base64 32`

Value	What it is	Generate with
`OFFLINE_GRANT_KEY`	EC P-256 private key (multi-line PEM) that signs the offline grants (step 4)	`openssl ecparam -genkey -name prime256v1 -noout`
`LICENSING_INSTALLATION_KEY`	one-time enrolment key from the licensing operator (`<id>.<secret>`); optional at start, without it apps cannot take a seat	issued by Software Notion

Generate the simple values and write them straight into `.env` (GNU `sed`; on macOS use `sed -i ''`):

```
sed -i "s|^POSTGRES_PASSWORD=.*|POSTGRES_PASSWORD=$(openssl rand -hex 24)|" .env
sed -i "s|^FOKUS_APP_PASSWORD=.*|FOKUS_APP_PASSWORD=$(openssl rand -hex 24)|" .env
sed -i "s|^SERVER_INSTANCE_ID=.*|SERVER_INSTANCE_ID=$(uuidgen)|" .env
sed -i "s|^SERVICE_CLIENT_SECRET=.*|SERVICE_CLIENT_SECRET=$(openssl rand -hex 24)|" .env
sed -i "s|^CERT_PASSWORD=.*|CERT_PASSWORD=$(openssl rand -hex 16)|" .env
sed -i "s|^BLOB_ENCRYPTION_KEY_BASE64=.*|BLOB_ENCRYPTION_KEY_BASE64=$(openssl rand -base64
32)|" .env
nano .env                   # organisation name, initial administrators, redirect URI, licence
key
```

3. Create certificates

The server signs and encrypts its tokens with two PKCS#12 certificates in `deploy/docker/keys/`, protected by `CERT_PASSWORD` from `.env`:

```
CERT_PASSWORD=$(grep '^CERT_PASSWORD=' .env | cut -d= -f2-)
mkdir -p deploy/docker/keys
T=$(mktemp -d)
openssl req -x509 -newkey rsa:2048 -nodes -keyout "$T/k.pem" -out "$T/c.pem" \
  -days 825 -subj "/CN=fokusfamilie"
openssl pkcs12 -export -inkey "$T/k.pem" -in "$T/c.pem" \
  -out deploy/docker/keys/signing.pfx    -passout pass:"$CERT_PASSWORD"
openssl pkcs12 -export -inkey "$T/k.pem" -in "$T/c.pem" \
  -out deploy/docker/keys/encryption.pfx -passout pass:"$CERT_PASSWORD"
rm -rf "$T"
chmod 644 deploy/docker/keys/*.pfx
```

The app container runs as an unprivileged user and must be able to read both files (`chmod 644`). For real deployments use your institutional PKI instead of a self-signed certificate.

4. Signing key for offline grants

`OFFLINE_GRANT_KEY` is an EC P-256 private key. The server signs the offline grants with it, so the apps can keep working without a connection and still verify their grants after a server restart. The key must stay stable and is backed up together with `.env`; a new key invalidates the offline grants issued so far.

Changing it is a planned rotation via `OFFLINE_GRANT_KEYSET_VERSION` and `OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY` (see "Rotating the offline grant keyset").

The value is a multi-line PEM; the whole PEM including the `BEGIN` / `END` lines goes into double quotes. Remove the empty line from the template and append the generated key:

```
sed -i '/^OFFLINE_GRANT_KEY=$/d' .env
{ printf 'OFFLINE_GRANT_KEY="'; openssl ecparam -genkey -name prime256v1 -noout; printf '"\n';
} >> .env
```

The result looks like this:

```
OFFLINE_GRANT_KEY="-----BEGIN EC PRIVATE KEY-----
MHcCAQEE...
-----END EC PRIVATE KEY-----"
```

Internal certificate authority (optional)

The server validates every outbound TLS connection (identity provider, `fflicense.software-notion.com`, version feed and update download) against the certificate authorities of the container. If a target uses a certificate from an **internal certificate authority** (typical for AD FS or Keycloak in the LAN) or a corporate proxy intercepts TLS (SSL inspection), put the issuing CA certificate in PEM format (`.crt` or `.pem` , several certificates per file allowed; DER `.crt` works too) into `deploy/docker/ca/` and restart the app:

```
cp internal-ca.crt deploy/docker/ca/
docker compose up -d --force-recreate app
docker compose logs app | grep "additional CA"     # Trusting 1 additional CA certificate(s)
...
```

`docker-compose.yml` mounts the directory read-only at `/etc/fokusfamilie/ca` (setting `OutboundTls__AdditionalCaDirectory`). These certificate authorities only apply when the normal validation fails solely because of the certificate chain; a certificate issued for another host name is always rejected. Revocation is not checked on this path (as with the default validation of the outbound connections; internal CAs often publish no reachable revocation list). The `updater` container does not need the certificates: it only talks to the Docker socket, the update download runs in the app.

5. First start and initial administrators

While the database contains no user yet, the server creates the initial administrators from `.env` at start-up; once users exist this step does nothing. No switch is needed:

```
BOOTSTRAP_ADMIN_EMAIL=admin@your-organisation.org
BOOTSTRAP_ADMIN_PASSWORD=<one-time password>
BOOTSTRAP_FACHADMIN_EMAIL=lead@your-organisation.org
BOOTSTRAP_FACHADMIN_PASSWORD=<one-time password>
BOOTSTRAP_PRIVACYADMIN_EMAIL=privacy@your-organisation.org
BOOTSTRAP_PRIVACYADMIN_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 initial admins. The bootstrap creates three separate accounts: the **technical admin** (system, devices, policies, licensing – no case or data-protection rights), the **professional admin** (cases, teams, practitioners, templates; may also sign in to the app) and the **data-protection admin** (data-subject requests and the privacy log). A pair left empty is skipped; at least `BOOTSTRAP_ADMIN_EMAIL` and `BOOTSTRAP_ADMIN_PASSWORD` are required for a new installation.

The initial passwords are **one-time passwords**: any value is accepted, the password rules do not apply to them. Like at sign-in, leading and trailing spaces and invisible control characters are ignored. Every account must replace its initial password at the first sign-in – in the panel as well as in the app – with a password that meets the rules; no other page is reachable before that. The initial password is **never** logged. Once the administrators have signed in, the `BOOTSTRAP_*` values can be removed from `.env`; they are ignored as soon as users exist.

Alternative: a single root admin. Small organisations where one person fills all three roles can create one root admin instead of three accounts: `BOOTSTRAP_ROOT_ADMIN=true` creates an account with system, professional and data-protection rights from `BOOTSTRAP_ADMIN_EMAIL` / `BOOTSTRAP_ADMIN_PASSWORD`; the professional and data-protection values are then ignored. The root admin must also replace the initial password at the first sign-in. The default is `false` (three separate accounts) because separating the roles strengthens data protection and accountability; rights can be split up later under **Users** at any time.

Start-up check. In production the server checks its configuration before it accepts requests: `OFFLINE_GRANT_KEY` present and readable, `BLOB_ENCRYPTION_KEY_BASE64` a 32-byte base64 key, both certificates present and readable with `CERT_PASSWORD`, `SERVER_INSTANCE_ID` a GUID, `ORGANIZATION_NAME` and the database password set, and – on an empty database – at least one initial administrator configured. If anything is missing, the app container exits with code 1 and lists every problem together with the command that fixes it:

```
docker compose logs app
# Focus Family cannot start: the production configuration is incomplete.
# Fix the following settings in .env (see .env.example) and run "docker compose up -d" again:
#   - OFFLINE_GRANT_KEY is empty. Generate it with: openssl ecparam -genkey -name prime256v1 -
noout ...
```

`BOOTSTRAP_ENABLED` from older installations is no longer needed and is ignored.

Back up `.env` (above all `CERT_PASSWORD`, `BLOB_ENCRYPTION_KEY_BASE64`, `OFFLINE_GRANT_KEY`, `SERVER_INSTANCE_ID`) and `deploy/docker/keys/*.pfx` separately from the data volumes. Without them attachments cannot be decrypted and devices cannot sign in after a restore.

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`,
- forwarded headers are only accepted from trusted proxy addresses. The compose network `web` has the fixed subnet `REVERSE_PROXY_NETWORK` (default `172.30.250.0/24`), which `docker-compose.yml` passes as `ReverseProxy__KnownNetworks__0`. A proxy container attached to `focusfamily_web` is therefore trusted automatically. Only the proxy and `app` should be attached to this network. Change the range only if it clashes with an existing network on the host (`docker compose up` then reports "Pool overlaps"),
- if the proxy runs **directly on the host** (forwarding to `127.0.0.1:8080`), requests arrive from the network's bridge gateway. Enter that address in `REVERSE_PROXY_IP`: `docker run --rm --network container:focusfamily-app-1 busybox ip route` → `default via <gateway>`,
- recommended: set `REVERSE_PROXY_ALLOWED_HOST` to the panel's host name (e.g. `focusfamily.your-organisation.org`). Forwarded headers are then only applied when `X-Forwarded-Host` carries this name, so a forged host never reaches redirects, the token issuer or setup QR codes. Let the proxy answer only for this name as well (no catch-all server). Left empty, the default `*` applies,
- choose a `client_max_body_size` (nginx) large enough for attachments,
- with nginx, enlarge the response header buffers: `proxy_buffer_size 16k; proxy_buffers 8 16k; proxy_busy_buffers_size 32k;` (included in the example). The default 4 KB buffer is not reliably large enough for single sign-on responses (cookies and redirect); nginx then answers "502 Bad Gateway" and logs `upstream sent too big header while reading response header from upstream`. Caddy has no such limit and needs no change.

Older compose files created the `web` network without a fixed subnet. After replacing the compose file, run `docker compose down` (without `-v`) and `docker compose up -d` once so Docker recreates the network; check with `docker network inspect focusfamily_web -f '{{range .IPAM.Config}}{{.Subnet}} {{end}}'`.

Development machines only

Two overlay files in the repository are meant for development and testing only and **never** belong on a server with real case data:

- `docker-compose.noclamav.yml` replaces ClamAV with a placeholder and switches off upload scanning (for machines without a ClamAV image, such as arm64).
- `docker-compose.local.yml` sets `ASPNETCORE_ENVIRONMENT=Local` and `Bootstrap__DevAdmin=true`. `admin@` then receives all permissions, no forced password change and the password from `.env` again at every start. The server only allows this in the `Development` and `Local` environments and additionally refuses it when `ReverseProxy:Enabled` is set, the licensing server points at `fflicense.software-notion.com` or the admin address does not belong to a development domain; in `Production` the setting is ignored with a warning in the log.

A local development stack starts with `docker compose --env-file .env -f docker-compose.yml -f docker-compose.noclamav.yml -f docker-compose.local.yml -p focusfamily up -d`.

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_ADMIN_EMAIL`, `BOOTSTRAP_ADMIN_PASSWORD`	initial technical admin; created automatically on the first start while the database has no user (one-time password, any value is accepted)
`BOOTSTRAP_FACHADMIN_EMAIL`, `BOOTSTRAP_FACHADMIN_PASSWORD`	initial professional admin (as above)
`BOOTSTRAP_PRIVACYADMIN_EMAIL`, `BOOTSTRAP_PRIVACYADMIN_PASSWORD`	initial data-protection admin (as above)
`BOOTSTRAP_ROOT_ADMIN`	`true` creates one root admin with all rights instead of the three accounts (default `false`)
`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_NETWORK`	fixed subnet of the compose network `web`; `X-Forwarded-*` from this range is trusted (default `172.30.250.0/24`)
`REVERSE_PROXY_IP`	optional additional proxy address, e.g. the bridge gateway for a proxy running directly on the host
`REVERSE_PROXY_ALLOWED_HOST`	optional host name of the panel; other `X-Forwarded-Host` values are ignored (default `*`)
`HTTPS_PROXY`, `HTTP_PROXY`, `NO_PROXY`	optional outbound proxy for the application and ClamAV (see "Outbound proxy")
`COMPOSE_PROFILES`, `FF_SELF_UPDATE`	`updater` and `true` enable updating from the admin panel (see "Updating from the admin panel")
`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)

Variable	Purpose
`Monitoring__ScrapeToken`	**licence server only:** access key for `/metrics` and `/monitoring/status`; empty = both addresses switched off
`Monitoring__AllowedNetworks`	**licence server only:** allowed IP addresses or CIDR ranges, comma separated; empty = any address with a valid key

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.

Monitoring: the admin server is **not** configured through the `.env` but in the panel under *System → Monitoring* — the access key lives there in the database instead of in clear text in a file. Only the licence server uses the two variables above. Details in "Operating and monitoring".

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.

Schema state 1.4.0: ledger version 38. The latest steps: 36 adds the column `Devices.DeviceModel` (device model, at most 64 characters), 37 the interface tables (`IntegrationSettings`, `IntegrationClients`, `IntegrationConnections`, `IntegrationCursors`, `IntegrationOutbox`, `IntegrationDeliveries`, `IntegrationDisclosures`, `IntegrationInbox`, `IntegrationExternalReferences`, `IntegrationTokens`), 38 the column `AspNetUsers."PermissionNames"`. Permissions are stored there as a set of names (`,CaseEdit,CaseView,`); step 38 converts the previous bitmask `"Permissions"` into names. The same reconciliation runs on every migrator start (permissions with a legacy bit follow the bitmask, permissions without one are kept), so that permissions granted or revoked on an older image after a rollback apply. If an account afterwards still holds permissions only as a bitmask, the migrator aborts and rolls back completely. The column `"Permissions"` keeps being written with the frozen legacy bits. Details: `docs/PERMISSIONS.md`.

Areas (ledger version 25): the tables `OrganizationAreaSettings` (switch per organisation; no row = off), `AdministrativeAreas` (unique name per organisation, case-insensitive), `AreaStaffAssignments` (a practitioner can belong to several areas, exactly one is the primary area; ledger version 27) and `AreaAdminGrants` (empty `AreaId` = organisation-wide; at most one organisation-wide row per account) plus the column `Teams.AreaId` (foreign key with `ON DELETE SET NULL`) are purely additive. No data is migrated; after the update area mode is off and behaviour is unchanged. The bulk GRANTs for `fokus_app` cover the tables.

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. Focus Family ships as a single instance.
- **Rollback:** restore the pre-upgrade backup into the previous image version.

From 1.0.0, 1.1.0 or 1.2.0 to 1.4.0: update the server first, then the apps; apps from version 1.0 on keep working against the updated server. The migrator applies the missing ledger versions: from 1.0.0 versions 22 to 38 (permissions `LicensePurchase`, `AreaManage` for practitioner admins and `SsoManage` for system administration, case timeline, activity record archive, areas, billing basis, single sign-on, invoice issuer and number series, policies "own Pro licence" and "help & contact", device model, interfaces, permissions as names), from 1.1.0 versions 34 to 38, from 1.2.0 versions 36 to 38; every step is additive, no manual SQL

step is needed. If you adopt this version's `docker-compose.yml`, recreate the `web` network once with `docker compose down` (without `-v`) and `docker compose up -d` (see "Set up the reverse proxy"). Once all devices are updated, set the **minimum client version** under **Policies** to `1.4.0`. Details: `docs/deploy/UPGRADE-1.0-TO-1.1.md`.

Updating from the admin panel

When the version feed reports a newer build, the overview page shows administrators holding **Configure system** the card **System update** with the button **Install update now**. The sequence is the same as above, only automated: the app downloads the image over TLS and verifies the SHA-256 checksum; the `updater` service (the only service with access to the Docker socket) backs up the database into the `backups` volume, loads the image, runs the migrator and restarts the app. If migration or start-up fails it restores the previous image automatically. Before installing, the updater reads version and build from the server assembly inside the signed image: if they differ from the request it aborts with `version_mismatch`; a build that is not newer than the running one is refused with `downgrade_refused` – also when it cannot determine the running build. A downgrade via the panel is therefore impossible. Data volumes are never touched; progress is visible in the panel and every step is recorded in the security audit.

Prerequisite: the `updater` service is running (`docker compose ps updater`) and the project directory is mounted at the same path as on the host (start the stack from inside the project directory or set `FF_PROJECT_DIR` in `.env`). The updater ships in the same image as the admin panel and only starts with a different entry point; after a successful update it replaces itself with the new version. Older compose files still build the updater from `deploy/docker/updater/` (`build:`); switch it to `image: fokusfamilie-server:<tag of app>` (template: current `docker-compose.yml`) and run `docker compose up -d` once.

The updater is optional, because it runs as root with access to the Docker socket. Two values in `.env` control it: `COMPOSE_PROFILES=updater` starts the service, `FF_SELF_UPDATE=true` makes the panel offer the installation. To run without it, remove `COMPOSE_PROFILES`, set `FF_SELF_UPDATE=false` and remove a running updater with `docker compose rm -sf updater`. The panel still reports new versions but only offers the download and notes that automatic updates are switched off; install them manually as described above. Without the service the button stays disabled and the panel says why. Log: `docker compose logs updater`.

Some updates are too extensive for the automated process or touch areas it does not cover. The panel then still shows **Update available** but offers only the image download; install it manually following the sequence above. A running process can only be reset in the panel as long as the updater has not taken it over yet, or once the service has sent no sign of life for ten minutes; a long-running step is shown as such.

Processor architecture: the server image is published for `linux/amd64` and `linux/arm64`; the version feed lists both downloads (`admin` and `admin_arm64`). The panel picks the image matching its own architecture, for the automated update as well as for the download link. If the feed has no image for this architecture, the panel shows **Update available** with a corresponding hint but starts no automated process; install manually once the matching image is published. The updater additionally refuses any image whose platform does not match the running container (`platform_mismatch`).

Beta versions: under **policies**, only an account with **system administration** can switch on "Show beta versions for updates" (off by default, column `OrganizationPolicies.AllowBetaUpdates`, ledger version 32). The version request to `api.software-notion.de` then contains `"includeBeta": true`, and the feed

may report pre-release versions; the switch takes effect immediately without waiting for the 24-hour cache and is recorded in the security audit. Recommended for test installations only.

Help & contact: the policy "Allow help & contact in the apps" (column `OrganizationPolicies.AllowSupportRequests`, ledger version 35, on by default) is distributed to the apps and desktop with the policy; older servers without the field count as "allowed". Requests from the panel are sent by the server over HTTPS to `https://software-notion.de/api/apps/focus-family/support-request` (override with `Support:Endpoint`); intranet installations without internet access use **Copy text** or **Send by e-mail** instead. The apps send directly from the device to the same address. Toggling is audited (`policy.support_requests_enabled` / `…_disabled`), sent requests as `support.request_sent` without the text.

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.

In the image, the directory `/var/lib/fokusfamilie/licensing` belongs to the app user (uid 1654). If the `licstate` volume is not writable anyway (for example root-owned), the server stores the identity in `dpkeys` as `licensing-installation.json` and logs a warning; on start it reads both locations, the configured path takes precedence. Before redeeming the key the server stores a pending enrolment (installation ID and key pair). If the connection drops or saving fails, the next attempt reuses the same identity and the licensing server confirms the repeat instead of rejecting the key as used. If an existing state file cannot be read, the server does not enrol again but shows an error.

Messages under Licensing:

Message (code)	Meaning and remedy
Key already used (`installation.enrollment_key_used`)	The key was redeemed by another installation. Ask the licensing operator for a new key or to reset the existing one, enter it in `.env`, restart `app`.
Key invalid / expired (`installation.enrollment_key_invalid`, `…_expired`)	Enter the complete key without spaces, or request a new key.
Timeout (`licensing.timeout`)	The licensing server does not answer in time; check firewall and outbound proxy.
TLS/certificate (`licensing.tls`)	Usually a TLS-intercepting proxy: put its CA certificate into `deploy/docker/ca/`.
Unreachable (`licensing.unreachable`)	DNS, connection or proxy: check name resolution, `HTTPS_PROXY` / `NO_PROXY` and firewall.
Licensing server or proxy error (HTTP 502/503/504)	Try again later; if it persists, check the outbound proxy.
Identity unreadable (`installation.state_unreadable`)	Check file permissions in `licstate` / `dpkeys` and make sure `dpkeys` and `encryption.pfx` belong to this installation.
Identity cannot be saved (`installation.state_write_failed`)	Neither `licstate` nor `dpkeys` is writable: check permissions (uid 1654) and disk space.

The log (`docker compose logs app | grep "Licensing:"`) records the cause with address, HTTP status, exception type and classification. Operator procedure to reset a used key: `docs/deploy/RUNBOOK.md` §6.

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.

Sponsored seats: the vendor can grant an organisation free seats in the licensing server (optionally with an end date). They belong to the organisation, not to the subscription, and never appear on an invoice. Capacity = purchased seats (with an active subscription holding at least one purchased seat) + sponsored seats; assignments are possible up to this number, even without a subscription. The licensing server's seat list separates `assignedUserIds` (all assignments) and `licensedUserIds` (effectively licensed: all with an active subscription, otherwise the earliest assignments up to the number of sponsored seats). The panel gates sync on `licensedUserIds` (a licensing server without the field: on the assignments) and shows remaining assignments as "Assigned, unlicensed"; the display and the reconciliation of orphaned seats use the assignments. Interfaces also count as licensed with sponsored seats only.

Compatibility of older panels: admin panels before 1.3.0 can already assign sponsored seats, because the licensing server checks the capacity, but only show the purchased number and do not know `licensedUserIds` (sync there follows all assignments).

Single sign-on (SSO)

Optionally staff sign in with their organization account – the same account they use for email, file storage and Windows sign-in. **OpenID Connect** (OAuth 2.0 authorization code flow with PKCE) and **SAML 2.0** are supported. This covers Microsoft Entra ID, on-premises Active Directory via AD FS, Keycloak, Okta, Google Workspace, Authentik and others. LDAP is not connected directly.

The server **federates**: apps (iOS, Android, desktop) and the admin panel keep signing in on this server's sign-in page. When SSO is active, the sign-in page additionally shows the button **Sign in with <name>**. The server redirects to the identity provider (IdP), validates its response and then issues its own tokens as usual. License seats, device binding, sync permissions and audit stay unchanged. The apps (iOS, Android, desktop) show their own button **Sign in with <name>** (without a configured name: **Sign in with organization account**) below the password sign-in while SSO is active. It opens the browser and leads directly to the identity provider – no email or password in the app. The apps learn about the active SSO from the server information (only "SSO on" and the display name, never addresses, domains or other settings) and query it again every time the sign-in view opens, so the button appears or disappears after a change in the panel without reinstalling. Apps of version 1.0 reach the same sign-in page with the SSO button via **Sign in with passkey**. If someone with an address of an allowed SSO domain enters email and password in the app, the app switches to the browser sign-in automatically.

Prerequisites

- The server is reachable at a **fixed HTTPS address** used by browsers and apps. This address is part of the redirect URI, ACS URL and entity ID. Behind a reverse proxy, `ReverseProxy:Enabled` must be set (see *Set up the reverse proxy*), otherwise the server generates wrong addresses.
- **Outbound HTTPS access** from the server to the identity provider: for OpenID Connect to the discovery document, signing keys (JWKS) and token endpoint; for SAML to the metadata URL – or you paste the metadata XML, then SAML needs no outbound access. The identity provider **never** needs to reach the server itself.
- A correct server **clock** (NTP): token and assertion lifetimes and sign-in re-confirmation are checked against time.
- A **local** account with the permission **"Manage single sign-on"** (`SsoManage`) and a password. Technical administration with system administration has this permission automatically. SSO accounts can never change the SSO settings – so a compromised identity provider cannot reconfigure sign-in.
- At least one **emergency access** account: an active local system admin account with a strong password and MFA (see *Emergency access and "Enforce SSO"*).
- Administrative rights at the identity provider to register an application.

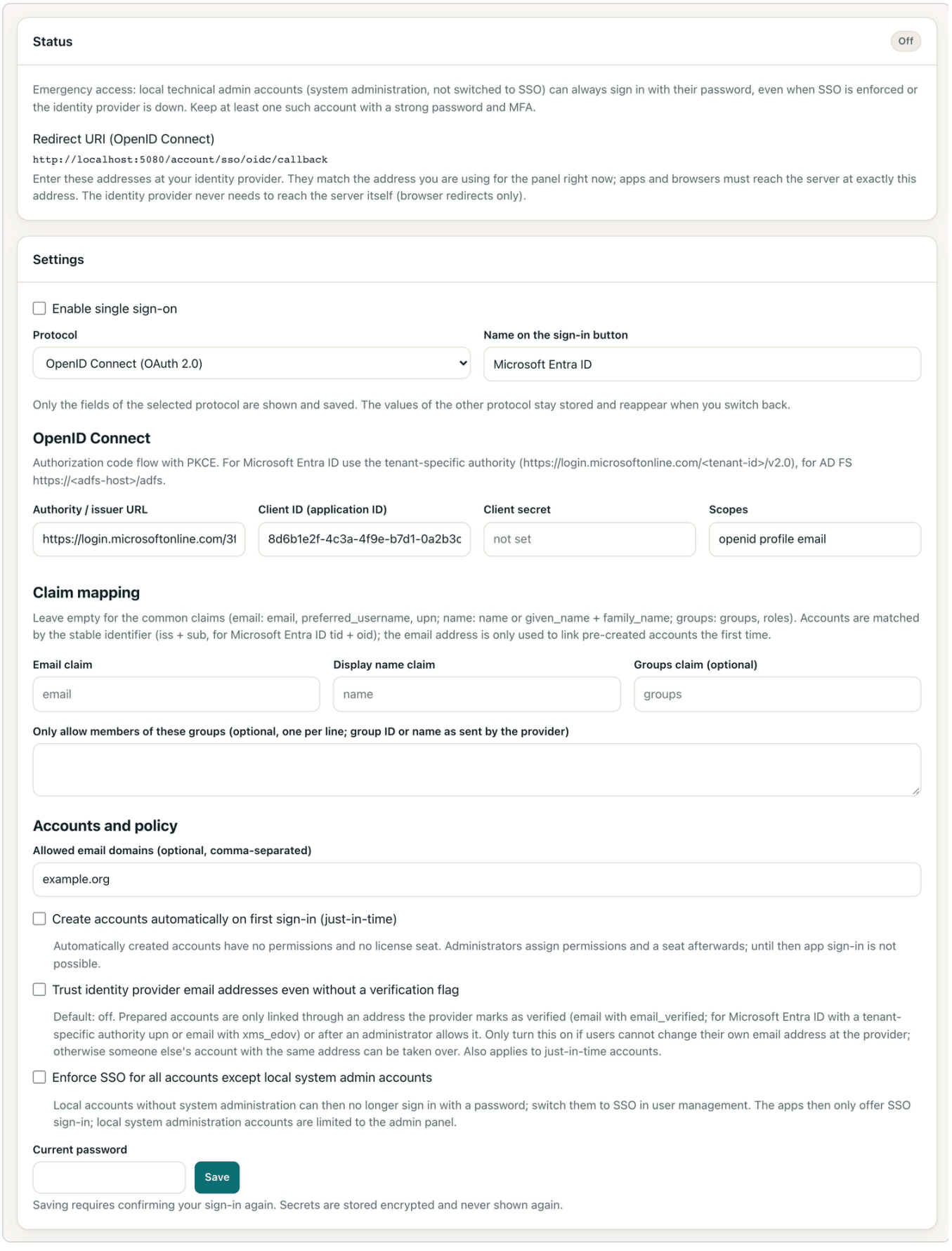
OpenID Connect or SAML 2.0?

Situation	Recommendation
Microsoft Entra ID, Keycloak, Okta, Google, Authentik	**OpenID Connect** – simpler to set up, email verification is included, pre-created accounts link without an extra step
On-premises Active Directory via AD FS	both work; **SAML** is the more common way with AD FS, OpenID Connect requires AD FS 2016 or newer
Google Workspace managed in the Admin console	**SAML** (app in the Admin console) or OpenID Connect via a Google Cloud project
Server without outbound internet access, IdP on the intranet only	**SAML** with pasted metadata XML (no outbound retrieval needed)
IT policy allows SAML applications only	**SAML**

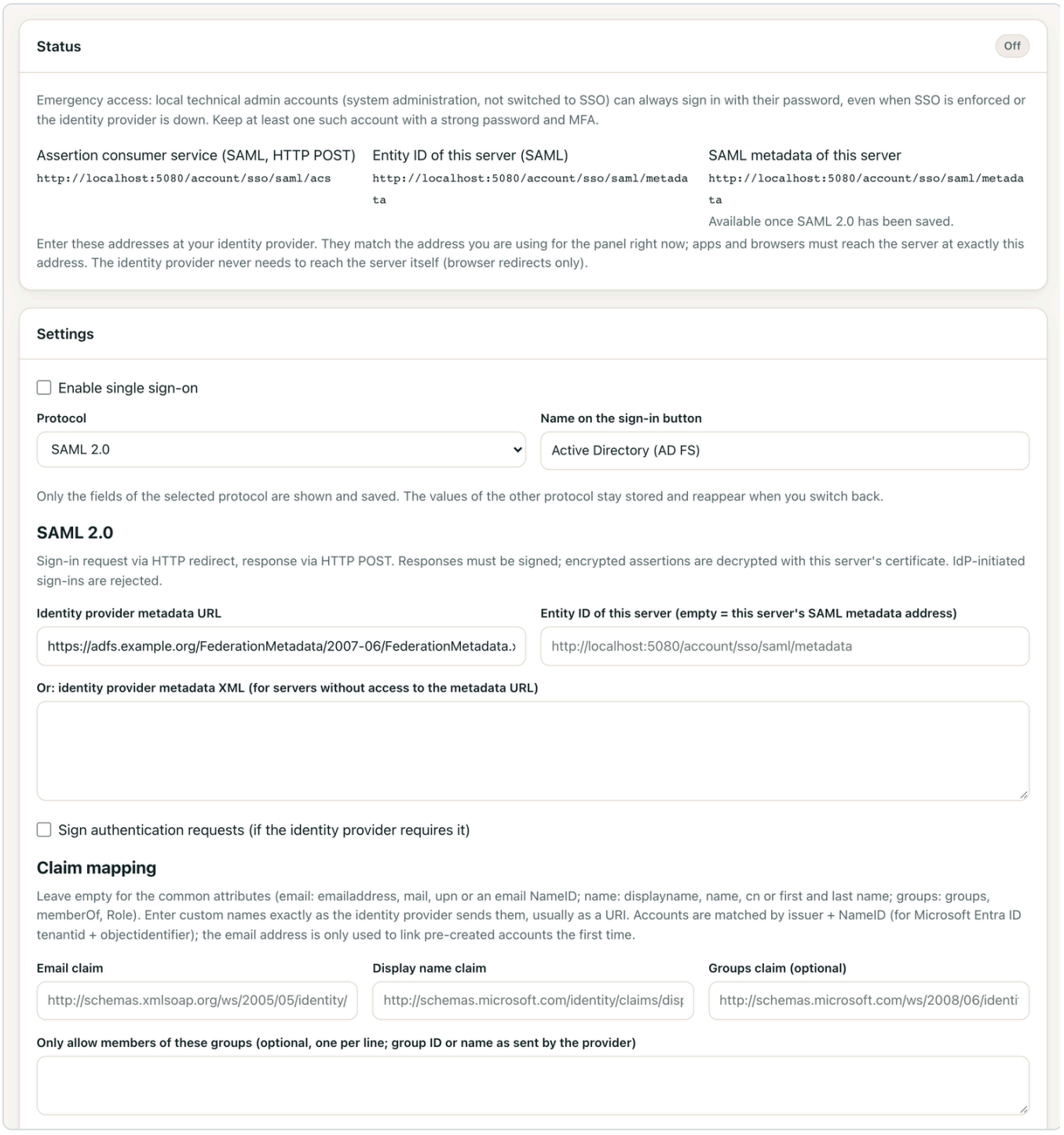
With SAML the identity provider never reports whether an email address is verified. Pre-created accounts are then only linked after **Allow first sign-in** or with the option **Trust identity provider email addresses even without a verification flag** (see *User accounts and linking*).

The "Single sign-on" page

Panel → **Single sign-on**. The page shows **only the fields of the selected protocol**. When you change the protocol in the selection, the page immediately shows the matching fields, addresses and hints. Only the fields of the selected protocol are saved; the values of the other protocol stay stored and reappear when you switch back. "Test connection" and the addresses refer to the **saved** settings – after switching the protocol, save first.



Single sign-on with OpenID Connect (example Microsoft Entra ID): redirect URI, authority, client ID, client secret and scopes.



Single sign-on with SAML 2.0 (example AD FS): ACS URL, entity ID and metadata of this server, identity provider metadata and SAML attributes. The "Accounts and policy" section is the same as for OpenID Connect.

Status and addresses

The badge shows **Active** (enabled and complete), **Enabled but incomplete** or **Off**. Below it are the addresses you enter at the identity provider. They match the address you are using for the panel right now – so open the panel via its official address, not via `localhost` or an IP address.

Protocol	Shown	Example
OpenID Connect	Redirect URI	`https://ff.example.org/account/sso/oidc/callback`
SAML 2.0	Assertion consumer service (HTTP POST)	`https://ff.example.org/account/sso/saml/acs`
SAML 2.0	Entity ID of this server	default `https://ff.example.org/account/sso/saml/metadata`
SAML 2.0	SAML metadata of this server (download link)	`https://ff.example.org/account/sso/saml/metadata`

This server's SAML metadata is only available once SAML 2.0 has been saved. It contains the entity ID, ACS URL and this server's certificate; many identity providers (AD FS, Keycloak, Entra ID) can create the application directly from it.

Common fields

Field	Meaning	Example
Enable single sign-on	Shows the SSO button on the sign-in page. When enabling, the server checks that the selected protocol is complete.	–
Protocol	OpenID Connect or SAML 2.0	OpenID Connect
Name on the sign-in button	Text of the "Sign in with …" button; empty = "SSO"	Microsoft Entra ID
Email / display name / groups claim	Only fill in if the provider uses names other than the usual ones (see *Claim mapping*). Applies to both protocols – check after switching the protocol.	empty
Only allow members of these groups	Optional, one group per line, exactly as the provider sends it in the groups claim (for Entra the object ID). No match: `sso_denied`.	`5b7c…-…-…`
Allowed email domains	Optional, comma-separated. Sign-ins with a different or missing email address are rejected.	`example.org, example.de`
Create accounts automatically on first sign-in (just-in-time)	Unknown people get an account without permissions and without a license seat.	off
Trust identity provider email addresses even without a verification flag	See *User accounts and linking* – mind the risk.	off
Enforce SSO for all accounts except local system admin accounts	Local password sign-in remains only for emergency access.	off
Current password (code or passkey if applicable)	Sign-in re-confirmation; without it nothing is saved.	–

OpenID Connect fields

Field	Meaning	Example
Authority / issuer URL	Base address of the provider; the server reads `<authority>/.well-known/openid-configuration`. HTTPS only.	`https://login.microsoftonline.com/<tenant-id>/v2.0`
Client ID (application ID)	Identifier of the application registered at the provider	`8d6b1e2f-4c3a-…`
Client secret	Client secret. Stored encrypted and never shown again; leaving it empty keeps the stored secret, **Delete the stored client secret** removes it.	–
Scopes	Space-separated; `openid` is always added. Default `openid profile email`.	`openid profile email`

The server always uses the authorization code flow with PKCE (`response_mode=query`) and validates signature, issuer, audience, lifetime and nonce. User data comes from the ID token; the UserInfo endpoint is not queried – required claims must be in the ID token.

SAML 2.0 fields

Field	Meaning	Example
Identity provider metadata URL	Retrieved and cached for one hour ("Test connection" reloads it). HTTPS only.	`https://adfs.example.org/FederationMetadata/2007-06/FederationMetadata.xml`
Or: identity provider metadata XML	For servers without access to the metadata URL. Must contain an HTTP redirect sign-in endpoint and a signing certificate. Paste it again when the provider changes its certificate.	`<EntityDescriptor …>`
Entity ID of this server	Identifier of this server at the provider (also "identifier", "audience", "relying party identifier"). Empty = this server's SAML metadata address.	empty
Sign authentication requests	Only enable if the provider requires signed AuthnRequests.	off

Sign-in requests go to the provider via HTTP redirect, the response arrives via HTTP POST at the ACS URL. The response or assertion must be signed with a certificate from the metadata; encrypted assertions are

decrypted with this server's certificate. Sign-ins started by the provider (IdP-initiated, for example from a tile in the Microsoft portal) are rejected – sign-in always starts on the Focus Family sign-in page. Use a **permanent NameID** format (persistent or an immutable identifier). A transient NameID changes with every sign-in and prevents matching (exception: Microsoft Entra ID, where `tenantid` + `objectidentifier` count).

Certificate of this server (SAML): when SAML is saved for the first time, the server creates a self-signed certificate (RSA 3072, valid for five years). It is used to sign requests and decrypt assertions and is included in this server's SAML metadata. **Create new certificate** (with sign-in confirmation) replaces it; afterwards register the new metadata at the provider, otherwise signed requests and encrypted assertions fail.

Claim mapping

Leaving the fields empty is almost always enough. The server then looks in this order:

Value	OpenID Connect	SAML 2.0
Email	`email`, `preferred_username`, `upn`, `unique_name`	`…/claims/emailaddress`, `email`, `mail`, `urn:oid:0.9.2342.19200300.100.1.3`, `…/claims/upn`, `upn`, otherwise a NameID in email format
Display name	`name`, `preferred_username`, otherwise `given_name` + `family_name`	`…/identity/claims/displayname`, `displayName`, `name`, `urn:oid:2.16.840.1.113730.3.1.241`, `cn`, otherwise first and last name
Groups	`groups`, `roles`	`…/claims/groups`, `groups`, `memberOf`, `Role`, `role`

If you enter a custom name, **only** that one is used. With SAML this is usually a URI, e.g. `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`. Group claims are only as trustworthy as their source: use group memberships managed by the provider, never attributes users can change themselves.

Test connection

Test connection checks the saved settings: for OpenID Connect, whether the discovery document can be retrieved and contains issuer, authorization and token endpoint; for SAML, whether the metadata can be retrieved or read and contains a valid signing certificate. Client ID and client secret are only checked by a real sign-in – so finish by testing with a test account in a private browser window.

Configuration via file or environment variables

If `Sso:Enabled` is set in `appsettings` or as an environment variable, the whole SSO configuration comes from the file and the panel only displays it:

```
Sso__Enabled=true
Sso__Protocol=oidc
Sso__DisplayName=Microsoft Entra ID
Sso__Oidc__Authority=https://login.microsoftonline.com/<tenant-id>/v2.0
Sso__Oidc__ClientId=<application-id>
Sso__Oidc__ClientSecret=<secret>
Sso__Oidc__Scopes=openid profile email
Sso__Saml__IdpMetadataUrl=https://adfs.example.org/FederationMetadata/2007-
06/FederationMetadata.xml
Sso__Saml__IdpMetadataXml=
Sso__Saml__SpEntityId=
Sso__Saml__SignAuthnRequests=false
Sso__Claims__Email=
Sso__Claims__Name=
Sso__Claims__Groups=
Sso__RequiredGroups=<group 1>;<group 2>
Sso__AllowedEmailDomains=example.org
Sso__JitProvisioning=false
Sso__TrustIdpEmail=false
Sso__EnforceSso=false
```

`Sso__Protocol` is `oidc` or `saml`. `Sso__AllowInsecureHttp=true` allows `http://` addresses for authority and metadata – for test environments only.

Microsoft Entra ID

OpenID Connect (recommended):

1. Entra admin center → **App registrations** → **New registration**: name "Focus Family", supported account types **Accounts in this organizational directory only**, platform **Web**, redirect URI from the panel (`https://<server>/account/sso/oidc/callback`).
2. **Overview**: note the *Application (client) ID* and *Directory (tenant) ID*.
3. **Certificates & secrets** → **New client secret**; copy the *Value* immediately (it is shown only once) and put the expiry date in your calendar.
4. **Token configuration** → **Add optional claim** → token type **ID**: `email`, `upn` and `xms_edov`. The `upn` claim is required to link pre-created accounts on first sign-in; `email` with `xms_edov: true` (domain verified by the tenant) is accepted as verified as well. For a group restriction also **Add groups claim** (security groups, emitted as group ID in the ID token).
5. **API permissions**: Microsoft Graph, delegated: `openid`, `profile`, `email`; **Grant admin consent**.
6. Optional: **Enterprise applications** → Focus Family → **Properties** → *Assignment required* = Yes, then assign the permitted people or groups under **Users and groups**.
7. In the panel: protocol **OpenID Connect**, name "Microsoft Entra ID", authority `https://login.microsoftonline.com/<directory-id>/v2.0` (**tenant-specific**, never `common`, `organizations` or `consumers`), client ID = application ID, client secret, scopes `openid profile email`. Save, **Test connection**, sign in with a test account, then **Enable single sign-on**.

8. Group restriction: enter the object IDs of the groups under **Only allow members of these groups** (one per line). If a person belongs to very many groups (more than 200), Entra sends no groups in the token – restrict via *Assignment required* instead.

Entra does not send `email_verified` for work accounts. Focus Family still accepts the address as verified if the token comes from a tenant-specific issuer (`https://login.microsoftonline.com/<tid>/v2.0`), the tenant ID `tid` matches the configured authority and the address comes from `upn` or from `email` with `xms_edov: true`. `preferred_username` never counts as verified. Entra accounts are matched by `tid` + `oid` – a later switch between OpenID Connect and SAML keeps the link. Renew the client secret in good time before it expires (create a new secret, enter it in the panel, delete the old one).

SAML:

1. **Enterprise applications** → **New application** → **Create your own application** ("non-gallery") → **Single sign-on** → **SAML**.
2. *Basic SAML Configuration*: **Identifier (Entity ID)** = entity ID from the panel, **Reply URL** = ACS URL from the panel. Alternatively *Upload metadata file* with this server's SAML metadata.
3. *Attributes & Claims*: the default claims (emailaddress, name, givenname, surname, objectidentifier, tenantid) are sufficient. For linking via email address, set the *emailaddress* claim to `user.userprincipalname`. For groups, **Add a group claim**.
4. *SAML Certificates*: copy the **App Federation Metadata Url** and enter it in the panel as metadata URL.
5. Assign **Users and groups**.
6. In the panel: protocol **SAML 2.0**, metadata URL, leave the entity ID empty (or use the identifier you entered), save, test, enable. Link pre-created accounts either with **Allow first sign-in** per account or enable **Trust identity provider email addresses even without a verification flag** – acceptable because users cannot change their own UPN in Entra.

On-premises Active Directory via AD FS

AD FS 2016 or newer publishes both SAML and OpenID Connect.

SAML (relying party trust):

1. AD FS Management → **Relying Party Trusts** → **Add Relying Party Trust** → *Claims aware* → **Import data about the relying party published online or on a local network** with `https://<server>/account/sso/saml/metadata` (or import the downloaded metadata file). AD FS then takes over entity ID, ACS URL and this server's certificate.
2. Choose an access control policy, for example *Permit specific group*.
3. **Edit Claim Issuance Policy** → rule *Send LDAP Attributes as Claims*, attribute store *Active Directory*: `E-Mail-Addresses` → *E-Mail Address*, `Display-Name` → *Name*, `Token-Groups – Unqualified Names` → *Group*.
4. Second rule *Transform an Incoming Claim*: *E-Mail Address* → *Name ID*, format **Persistent Identifier**. Alternatively use an immutable identifier (e.g. `objectGUID`) as persistent NameID.
5. In the panel: protocol **SAML 2.0**, metadata URL `https://<adfs-host>/FederationMetadata/2007-06/FederationMetadata.xml` (without outbound access: open the file and paste the XML), leave the groups claim empty or use `http://schemas.microsoft.com/ws/2008/06/identity/claims/groups`, enter the group names under **Only allow members of these groups**. Since the addresses come

from the directory and users cannot change them, **Trust identity provider email addresses even without a verification flag** can be enabled.

OpenID Connect:

1. AD FS Management → **Application Groups** → **Add Application Group** → template **Server application accessing a web API**.
2. Server application: redirect URI from the panel, note the **Client Identifier**, under *Configure Application Credentials* **Generate a shared secret** and copy it.
3. Web API: enter the client identifier as identifier; choose an access control policy; permitted scopes `openid`, `profile`, `email` and `allatclaims`.
4. In the web API, under **Issuance Transform Rules**, add a rule *Send LDAP Attributes as Claims*: `E-Mail-Addresses` → *E-Mail Address*, `User-Principal-Name` → *UPN*, `Display-Name` → *Name*, if needed `Token-Groups – Unqualified Names` → *Group*.
5. In the panel: protocol **OpenID Connect**, authority `https://<adfs-host>/adfs`, client ID = client identifier, client secret, scopes `openid profile email allatclaims`. AD FS does not send `email_verified`: link pre-created accounts with **Allow first sign-in** or enable **Trust identity provider email addresses even without a verification flag**. For groups, enter the groups claim `group` if AD FS names it that way.

Keycloak

- **OpenID Connect:** realm → **Clients** → **Create client**, type *OpenID Connect*, client ID e.g. `focus-family`; *Client authentication* **on**, *Standard flow* **on**, all other flows off; **Valid redirect URIs** = redirect URI from the panel; under *Advanced* → *Proof Key for Code Exchange* = `S256`. Copy the client secret under **Credentials**. Groups: **Client scopes** → `focus-family-dedicated` → **Add mapper** → *Group Membership*, token claim name `groups`, *Full group path* off, *Add to ID token* on. In the panel: authority `https://<keycloak>/realms/<realm>`, client ID, secret. Keycloak sends `email_verified`; make sure email addresses in the realm are verified (*Realm settings* → *Login* → *Verify email*) and users cannot change their address without verification.
- **SAML: Create client**, type *SAML*, client ID = entity ID from the panel; **Valid redirect URIs** and *Assertion Consumer Service POST Binding URL* = ACS URL; *Name ID format* `persistent`; *Sign assertions* on; *Client signature required* only on if **Sign authentication requests** is enabled in the panel (then import this server's certificate under *Keys*). Mapper *User Property* `email` → SAML attribute `email`. Metadata in the panel: `https://<keycloak>/realms/<realm>/protocol/saml/descriptor`.

Okta

- **OpenID Connect: Applications** → **Create App Integration** → *OIDC – OpenID Connect*, *Web Application*; **Sign-in redirect URIs** = redirect URI from the panel; grant type *Authorization Code*; assign people or groups. In the panel: authority `https://<org>.okta.com` (or a custom authorization server, e.g. `https://<org>.okta.com/oauth2/default`), client ID, client secret. Groups: under *Sign On* → *OpenID Connect ID Token* → *Groups claim* set a filter (claim `groups`).
- **SAML: Create App Integration** → *SAML 2.0*; *Single sign-on URL* = ACS URL, *Audience URI (SP Entity ID)* = entity ID, *Name ID format* `Persistent`, attribute `email` = `user.email`, optionally *Group Attribute Statement* `groups`. Enter the app's *Metadata URL* in the panel.

Google Workspace

- **SAML (Admin console): Apps → Web and mobile apps → Add app → Add custom SAML app**. In the *Google Identity Provider details* step, **Download metadata**. Under *Service provider details*: **ACS URL** and **Entity ID** from the panel, *Name ID format* `EMAIL`, *Name ID = Basic Information › Primary email*. Attribute mapping: *Primary email* → `email`, *First name* → `givenName`, *Last name* → `sn`. Turn the app on under **User access** for the desired organizational units. In the panel: protocol **SAML 2.0**, paste the downloaded metadata XML. Google uses the primary email address as NameID; if it is renamed, the identifier changes (then **Remove link** and link again). Since the addresses are managed in the Admin console, **Trust identity provider email addresses even without a verification flag** can be enabled.
- **OpenID Connect (Google Cloud):** project → **APIs & Services** → **OAuth consent screen** (type *Internal*) → **Credentials** → **OAuth client ID** → *Web application*, **Authorized redirect URIs** = redirect URI. In the panel: authority `https://accounts.google.com`, client ID, client secret, and always set **Allowed email domains** to your own domain (otherwise any Google account for which an account is prepared could sign in).

Other providers

Every OpenID Connect provider with a discovery document (`<authority>/.well-known/openid-configuration`) and authorization code flow, or every SAML 2.0 provider with metadata, an HTTP redirect sign-in endpoint and signed responses works (e.g. Authentik, Shibboleth, PingFederate, JumpCloud). In general: register the application as a confidential web client or service provider, enter the addresses from the panel, emit email address and name in the ID token or as attributes, and choose a permanent NameID for SAML.

User accounts and linking

Type	Sign-in	Password, MFA, passkeys
Local account	email and password (MFA or passkey if applicable)	managed in the panel
SSO account (badge **SSO**)	only through the identity provider	at the identity provider; blocked in the panel and the API

Matching: an SSO account is permanently matched by the provider's identifier – OpenID Connect `iss` + `sub`, SAML issuer + NameID, Microsoft Entra ID in both protocols `tid` + `oid`. The email address is only used for the **first link**; later address changes at the provider do not matter.

Pre-create accounts (recommended): Users → **New SSO account** (name, email address or UPN, permissions, area). You can assign permissions and a license seat right away. Until the first sign-in the badge shows **pending**. On first sign-in the server links the account via the email address – but only if it counts as **verified**:

Provider	Counts as verified
OpenID Connect in general	claim `email` together with `email_verified: true`
Microsoft Entra ID (OpenID Connect)	`upn` or `email` with `xms_edov: true`, only with a tenant-specific authority and matching `tid`
SAML 2.0	never – SAML has no verification flag
Option **Trust identity provider email addresses even without a verification flag**	every address delivered by the provider

If the address is not verified, the server rejects the sign-in with `sso_email_unverified`. An administrator with user management then allows the first sign-in: panel → **Single sign-on** → **Accounts without organization sign-in yet** → **Allow first sign-in** (area-bound administration uses the user page for this). The next sign-in with a matching email address within **24 hours** is linked **once**. Only allow it when the person is about to sign in. The permission expires on linking, on removing the link and on switching the sign-in method.

Trust identity provider email addresses even without a verification flag (default off) treats every address from the provider as verified – for the first link and for just-in-time. Only enable it if users **cannot change** their email address at the provider themselves (AD FS with addresses from Active Directory, Entra SAML with `user.userprincipalname`, Google Workspace). **Risk:** if someone can choose their own address at the provider (self-service profile, social login, Keycloak without email verification), they take over someone else's prepared account including its permissions.

Just-in-time (default off) creates unknown people on first sign-in as practitioners **without permissions and without a license seat**; until an administrator assigns both, app sign-in is not possible. Just-in-time also requires a verified address (or the trust option). Without just-in-time unknown people get `sso_unknown`.

Allowed email domains and groups restrict every sign-in. For an already linked account with an unverified address, the domain of the address stored in the account must be allowed.

License: license seats are assigned to SSO accounts exactly like to local accounts (panel → Licensing).

Switching the sign-in method and removing the link

In a person's menu under **Sign-in method** (user management permission; technical accounts additionally require "Manage technical admins"):

- **Switch to SSO:** password, MFA key, passkeys and open setup codes are removed, all sessions and device sign-ins end. Permissions and license seat are kept. The email domain must be allowed and SSO must be active. A local account with the same email address is **never taken over automatically** (message `sso_local`), only switched this way.
- **Switch to local sign-in:** removes the link and ends all sessions; set a password afterwards.
- **Remove link:** only removes the link (for example after changing the provider). The next sign-in links again – through a verified address or after **Allow first sign-in**.
- You cannot switch your own sign-in method.

Switching protocol or provider: the identifier depends on protocol and provider. With Microsoft Entra ID it stays the same when switching between OpenID Connect and SAML (`tid` + `oid`). With all other providers the identifier changes (`iss` + `sub` versus issuer + NameID); linked accounts then get `sso_conflict`. In

that case remove the links (**Remove link**) before or after the switch and let them link again. Also check the claim mapping; it applies to both protocols.

Disabling SSO: SSO accounts stay blocked (fail-closed) until they are switched to local sign-in.

Emergency access and "Enforce SSO"

- Local accounts with **system administration** that are not switched to SSO **always** sign in to the **admin panel** with their password – even with SSO enforced or the identity provider down. Emergency access is a panel-only path: with SSO enforced, such an account also reaches the apps only via SSO; password and passkey sessions receive no app tokens, and existing app sign-ins end at the next token renewal. Keep at least one such account with a strong password and MFA and store its credentials safely (vault, four-eyes principle).
- The last active local system admin account with a password cannot be switched to SSO. Without such an account **Enforce SSO** cannot be enabled; the page then shows a warning.
- **Enforce SSO** blocks password sign-in for all other local accounts (panel, app, API: `sso_only`). Switch these accounts to SSO beforehand. With SSO enforced, the apps no longer show email/password fields or the passkey button in server connection, sign-in again and credential unlock – only **Sign in with …** and the hint "Sign-in uses the organization account." Without enforcement the form stays, with the additional SSO button. Switching it on ends the existing local sessions and app tokens of these accounts immediately.
- **Password attempts with an address of an allowed SSO domain** (field "Allowed email domains", SSO active) always answer `sso_only` – whether the account exists, is SSO-managed, deactivated or blocked, or the password is wrong. The sign-in page and apps thus guide everyone of these domains to the SSO sign-in without revealing whether an account exists. Only accounts that are not blocked (emergency access, or local accounts without "Enforce SSO") sign in with their correct password. For addresses outside these domains, and while SSO is off or no domains are entered, the generic error appears as before; `sso_only` then only follows a correct password.
- For SSO accounts, password sign-in, setting and changing passwords, MFA setup, passkeys and setup QR codes are blocked on the server. The global MFA requirement counts as fulfilled; enforce MFA at the identity provider (e.g. Entra conditional access policies).

Sign-in re-confirmation

Protected actions require a fresh sign-in confirmation. Local accounts confirm with their password (code or passkey if applicable). SSO accounts (for example in area mode) confirm via **Sign in again now**: the server requests a complete new sign-in at the provider – for OpenID Connect with `prompt=login` and `max_age=0`, evaluating the `auth_time` claim; for SAML with `ForceAuthn="true"`, evaluating `AuthnInstant`. The reported authentication time must be at most **5 minutes** old and not older than the start of the confirmation (1 minute clock tolerance); the confirmation is then valid for **10 minutes**. A normal SSO sign-in never counts as confirmation. If the provider does not deliver `auth_time` or `AuthnInstant`, or ignores the new sign-in (for example because of an existing session), `sso_reauth` appears.

Intranet operation

SSO works exclusively with **browser redirects** (front channel): the browser is sent to the provider and returns with the result. The provider never calls the server directly, so panel and server can be reachable only on the internal network or via VPN. The browser must reach both.

- **OpenID Connect:** the server needs outbound HTTPS to the provider's discovery document, JWKS and token endpoint (for Entra `login.microsoftonline.com`). An outbound proxy is used as described under *Outbound proxy*.
- **Internal certificate authority:** if the provider uses a certificate from an internal CA, put that CA certificate into `deploy/docker/ca/` (see *Internal certificate authority* in the installation chapter). Otherwise sign-in fails with `sso_idp`, the log reports "certificate chain not trusted - add the issuing CA to deploy/docker/ca" and **Test connection** shows the same reason.
- **SAML:** with pasted metadata XML the server needs **no** outbound access – suitable for isolated networks with an internal AD FS.
- There is no single logout: signing out of the panel only ends the Focus Family session, not the one at the provider.

Security and secrets

- The client secret and the private key of the SAML certificate are stored encrypted with the Data Protection key ring (volume `dpkeys`) in the table `SsoConfigurations` and are never displayed or logged. If the key ring is lost, enter the secret again and create a new SP certificate.
- Every change to the SSO settings requires the sign-in re-confirmation of a local account with the permission **"Manage single sign-on"** (`SsoManage`). This permission is separate from "System configuration"; per-person actions (switch, remove link, allow first sign-in) belong to user management (`UserManage`).
- SAML responses are checked for signature, issuer, audience (entity ID), lifetime, `Destination`/`Recipient`, `InResponseTo` (only responses to the server's own requests at most 10 minutes old) and replay; the response is bound via an HttpOnly cookie to the browser that started the sign-in.

SSO troubleshooting

Messages on the sign-in page (code in the `error=` parameter of the address):

Code	Meaning	Remedy
`sso_only`	The account signs in via SSO only, SSO is enforced, or the address belongs to an SSO domain.	Use the **Sign in with …** button (in the apps below the password sign-in); apps of version 1.0: **Sign in with passkey** opens the same page. Emergency access accounts of an SSO domain: check the password.
`sso_disabled`	SSO is off or incomplete.	SSO page: check status, enable.
`sso_idp`	Provider unreachable (TLS certificate not trusted, DNS, connection, timeout) or SAML metadata not retrievable or without a valid signing certificate.	**Test connection** shows the reason; for an internal CA put its certificate into `deploy/docker/ca/` and restart the app; otherwise check network/proxy or paste the XML.
`sso_claims`	The provider sends no identifier (`sub`/NameID) or no email address.	Emit the claims or attributes at the provider; set the claim mapping if needed.
`sso_denied`	Domain or group not allowed, or account deactivated.	Check allowed domains, groups, groups claim and account status.
`sso_unknown`	No account exists and just-in-time is off.	Create the account as **New SSO account** or enable just-in-time.
`sso_local`	Local password account with the same email address.	Switch the account to SSO under **Sign-in method**.
`sso_conflict`	The account is linked to a different identifier (provider or protocol switch, transient NameID).	**Remove link**; for SAML configure a permanent NameID.
`sso_email_unverified`	Address not reported as verified.	**Allow first sign-in**, check provider claims (Entra: `upn`, tenant-specific authority) or deliberately enable the trust option.
`sso_reauth`	Sign-in confirmation not recent (`auth_time` / `AuthnInstant` missing or too old).	Sign in again completely at the provider; check the clocks of server and provider.
`sso_nameid`	The SAML provider sent a transient NameID.	Configure the NameID format `persistent` for Focus Family at the provider.
`sso`	Invalid response (signature, lifetime, replay, foreign or expired request, wrong client secret).	Check the server log (`docker compose logs app`) and redirect URI or ACS URL.

Messages when saving: *valid HTTPS address as authority*, *enter the client ID*, *valid HTTPS address for the metadata*, *enter metadata URL or metadata XML*, *metadata XML invalid* (no HTTP redirect endpoint or no signing certificate), *enter valid domains*, *SSO cannot be enforced* (no emergency access), *confirming your sign-in failed*.

Other common causes: redirect URI or ACS URL at the provider differs (scheme `http` instead of `https` behind the proxy, different host name, missing path); expired client secret; certificate change at the provider with pasted metadata XML; sign-in from a tile in the provider portal (IdP-initiated, rejected).

Audit events

Event	Meaning
`sso.config_changed`	settings saved (field names only, never values) or saving rejected without valid confirmation
`sso.connection_tested`	"Test connection" with result
`sso.sp_certificate_rotated`	new SAML certificate created
`sso.login`	successful SSO sign-in (reason `oidc`, `saml`, possibly `_reauth`)
`sso.login_denied`	rejected sign-in with reason, e.g. `sso:email_unverified`, `sso:domainnotallowed`, `sso:reauth_stale`
`sso.user_linked`	first link (reason `verified_email` or `link_grant`)
`sso.user_provisioned`	account created via just-in-time
`sso.password_login_rejected`	password sign-in of an SSO account or with SSO enforced rejected (including app sign-in of an emergency-access account without SSO)
`sso.app_token_rejected`	issuing or renewing app tokens without an SSO sign-in rejected while SSO is enforced
`user.sso_enabled` / `user.sso_disabled`	account switched to SSO or local
`user.sso_unlinked`	link removed
`user.sso_link_granted`	first sign-in allowed

SSO rollout checklist

1. Check emergency access (local system admin account with password and MFA).
2. Register the application at the provider, take over the addresses from the panel.
3. Fill in the panel fields, save, **Test connection**.
4. Create a test account as **New SSO account**, sign in in a private browser window, check the audit.
5. **Enable single sign-on**, pre-create or switch accounts, assign license seats.
6. Only then, if desired, **Enforce SSO**.
7. Put the expiry date of the client secret or the provider certificates in your calendar.

Interfaces

The server connects case management systems (REST/JSON API and webhooks), document management and e-files (CMIS in both directions), XÖV recipients (xdomea via directory, SFTP, XTA 2 or FIT-Connect), identity providers (SCIM 2.0), work calendars (iCalendar subscriptions) and SIEM (syslog). Setup, reference, privacy defaults, operation and troubleshooting are described in the **Interfaces handbook**. In the panel it is

available under **Interfaces** → **Interface documentation**, together with the OpenAPI description of the REST API.

The essentials for operation:

- **Permissions:** setup requires **Manage interfaces (REST clients, CMIS, XÖV, webhooks, SCIM, SIEM)**; the permission is included in system admin. Data flows carrying case data only start once **another** person with the **Data protection officer** permission approves them with a local account. The page **Incoming (interfaces)** requires the operational permission **Review and accept interface inbox items**, assigning and accepting additionally **Edit cases**.
- **Addresses:** REST API `/api/integration/v1`, token endpoint `/connect/token`, CMIS repository `/cmis/browser`, SCIM `/scim/v2`, calendar subscriptions `/feeds/calendar/<token>.ics`. The supplied reverse proxy templates forward all paths; if you allow paths individually, add these. The IP lists of API clients need the settings from *Set up the reverse proxy*, otherwise the server only sees the proxy's address.
- **Outbound connections:** the interfaces' target systems add to the table under *Outbound connections of the server*. Internal networks are allowed; loopback, link-local and metadata services are blocked. Certificate authorities from `deploy/docker/ca/` apply here too. Behind an outbound proxy, internal target systems belong in `NO_PROXY`.
- **XÖV via directory:** the root directory `/var/lib/fokusfamilie/xoev` is not a volume in the standard stack. For the hand-over to a gateway, mount a host directory (owner UID 1654).
- **Secrets** of the connections are stored in the database encrypted with Data Protection. The volume `dpkeys` therefore belongs in the backup; without these keys all secrets and client certificates of the interfaces must be entered again.
- **Monitoring:** figures `fokusfamilie_integration_*` under `/metrics`, audit events `integration.*`, **Transfer log** in the panel. Emergency stop: **Pause all outgoing transfers (emergency stop)**.
- **Licence:** without a valid organisation licence (neither an active subscription with purchased seats nor sponsored seats) deliveries rest and the interfaces refuse requests; after the next successful licence check work continues without data loss. The check result is cached for five minutes. If the licensing server cannot be reached, a previously positive result stays valid for up to 24 hours; a refusal by the licensing server counts as "not licensed" immediately.

Data room

Under **Data room** (permission **Manage data room**, included in the system admin) you provide practitioners with forms, templates and service instructions from a network drive (NAS) or a WebDAV server. The server accesses the source; the app and the desktop show the folders under **Data room** and open documents read-only. Devices need no access to the network drive and no VPN into the file network.

- **Create a source:** name, type (*Network drive* or *WebDAV*), folder or address, for WebDAV user name and password. A dedicated read-only account is recommended. The password is stored encrypted with Data Protection in the database (back up the `dpkeys` volume) and never sent to devices. When the address or user name changes, the password must be entered again.
- **Network drive:** the share is mounted read-only on the server and passed to the container as `/dataroom` (setting `DataRoom__Root`, instructions in the operations runbook §4h). Without a mount the page shows a notice.

- **Visibility:** without a team selection all practitioners see the source, with a selection only members of those teams. *Available to practitioners* hides a source without deleting it.
- **Offline copies:** with *Allow offline copies on devices* practitioners can keep single documents for use on the go. They are stored encrypted in the app, protected by the app lock, and removed at the next refresh when the source is deleted, disabled or no longer shared with them.
- **Test and log:** *Test connection* reads the top level and reports the number of entries or the cause (folder missing, sign-in refused, certificate, blocked address, unreachable). Creating, changing, testing and deleting are logged as `dataroom.*` in the audit log, every opened document as `dataroom.file_opened` with source and path.
- **Protection:** hidden files, recycle bins and thumbnail folders of NAS systems as well as symbolic links are not shown; paths with `..` or drive letters are refused. WebDAV uses HTTPS (http only when explicitly allowed for the internal network) with the same address rules as the interfaces.

Operation on the intranet or via VPN

The admin panel can be reachable only on the internal network or via VPN; the reverse proxy then publishes the host name only in internal DNS and accepts connections only from the internal network and the VPN address range. The server needs no inbound connection from the internet: the licensing server, the payment provider and the version feed never call the panel. Outbound, however, the server must reach **`fflicense.software-notion.com` over HTTPS (port 443)**.

Outbound connections of the server

Destination	Purpose	Required?	When blocked
`fflicense.software-notion.com:443`	First enrolment of the installation, seat check on sync, managing seats and licensing devices	**yes**	Without the first enrolment devices cannot check their licence. Afterwards, users already confirmed keep syncing until the panel restarts; then sync pauses until the licensing server is reachable again. The apps stay silently offline, no data is lost.
`api.software-notion.de:443`	"Update available" notice	no	no update notice
`static.software-notion.de:443`	Download of the update from the panel	no	install the update manually
`database.clamav.net:443`	Virus signatures	recommended	Signatures go stale; without initial signatures ClamAV, and therefore the application, does not start. Copy the signature files into the `clamdb` volume once in that case.
internal time server (NTP)	accurate clock	**yes**	sign-in and licence checks fail with clock skew
identity provider (SSO only)	OpenID Connect: discovery, JWKS and token endpoint; SAML: metadata URL	SSO only	SSO sign-in fails; with pasted SAML metadata XML no access is needed

Inbound, allowing port 443 of the reverse proxy from the internal network and the VPN address range is enough. Handbook links and the "Manage seats" page are opened by the administrator's browser; that workstation needs internet access, the server does not.

Outbound proxy

For a proxy, add to `.env`:

```
HTTPS_PROXY=http://proxy.your-organisation.internal:3128
HTTP_PROXY=http://proxy.your-organisation.internal:3128
NO_PROXY=localhost,127.0.0.1,db,clamav
```

The application and ClamAV then use the proxy automatically (`docker compose up -d --force-recreate app clamav`). `NO_PROXY` must contain the internal services `db` and `clamav` as well as `localhost`. Exclude `fflicense.software-notion.com` from TLS inspection. Check: `docker compose exec app curl -sS -o /dev/null -w "%{http_code}\n" https://fflicense.software-notion.com/health/live` returns `200`.

Devices, licence and sync

- The apps sync as soon as they reach the panel on the internal network or via VPN. Outside it they keep working fully offline, without error messages and without a lock; changes are transferred on the next contact.
- Each device checks its licence **itself, directly with the licensing server**, over its own internet connection. The panel only issues a short-lived confirmation for this. For a successful check the device must therefore reach the panel and `fflicense.software-notion.com` at the same time, for example on the internal network with internet access or through a split-tunnel VPN.
- If the check does not succeed, the grace period applies: days 0–3 silent, days 3–5 one notice per day, days 5–10 hourly, from day 10 locked until the next successful check. Devices should therefore be connected at least every few days.
- The "Max offline days" policy should match this usage. When it expires, the app only shows a notice to sign in again.

DNS and certificates

- Use the **same host name** on the LAN and the VPN (split-horizon DNS, or the internal DNS server also inside the VPN). Sign-ins are bound to the address; changing the name forces a new sign-in.
- Open the panel under exactly this name when creating QR codes; the QR code takes over the current address.
- The mobile apps connect over HTTPS only, and the server issues sign-ins over HTTPS only. The simplest option is a publicly trusted certificate for an internal name under your domain, issued through the Let's Encrypt DNS challenge (DNS-01).
- A certificate from your internal certificate authority works on iOS/iPadOS (distribute the root certificate in an MDM profile, or enable it under Settings → General → About → Certificate Trust Settings) and in the desktop apps (the operating system's certificate store). **The Android app trusts system certificate authorities only**; it does not accept user certificates or certificates installed by MDM. Android devices therefore need a publicly trusted certificate.

VPN

- **Split tunnel:** internal network through the tunnel, internet direct. The simplest option.
- **Full tunnel:** possible if the internal network offers an internet exit to `fflicense.software-notion.com`.
- **Always-on VPN** (iOS via MDM, Android "Always-on VPN", Windows Always On VPN) keeps sync and licence up to date without user action.
- **Per-app VPN:** assign Focus Family to the VPN; for sign-in through the system browser also the browser or the panel's domain if needed.

Detailed tables and check steps are in the operations runbook (`docs/deploy/RUNBOOK.md`, section 4).

Distributing the apps through device management

Phones, tablets and computers that the organisation manages with a device management system (MDM, for example Microsoft Intune, Workspace ONE, Jamf, Ivanti or Samsung Knox Manage) get Focus Family as

a public app from the official stores. The organisation needs neither its own store listing nor its own developer account.

Platform	Distribution	Identifier	Minimum version
Android (phone, tablet)	Managed Google Play (Android Enterprise)	Package name `com.softwarenotion.focusfamily`	Android 9
iPhone, iPad	Apple Business Manager, *Apps and Books*	"Focus Family", App Store ID `6799945064`	iOS/iPadOS 17
Windows	MSI package as a Win32 or line-of-business app	`FocusFamily-1.4.0-x64.msi` or `-arm64.msi`	Windows 10 22H2
macOS	Signed and notarised PKG package	`FocusFamily-1.4.0-AppleSilicon.pkg` or `-Intel.pkg`	macOS 14
Linux	The organisation's package distribution	`.deb` and `.rpm` for amd64 and arm64	Ubuntu 22.04, Debian 12, Fedora 42

The desktop packages are available on the admin panel dashboard and at `https://static.software-notion.de/focusfamily/`, each with checksums (`SHA256SUMS`).

What the app does not support: Focus Family contains no device management SDK. App configuration through device management (Android "managed configurations", iOS "Managed App Configuration") and app protection policies without device enrolment (for example Intune MAM) therefore have no effect. Distribution, updates and device policies run through device management; practitioners connect the app to the server as described below.

Android through Managed Google Play

Using Microsoft Intune as the example; other device management systems with Android Enterprise follow the same steps.

1. Connect the Intune account to Managed Google Play once: *Tenant administration → Connectors and tokens → Managed Google Play*.
2. *Apps → Android → Add → Managed Google Play app*, search for "Focus Family", choose **Approve** and confirm the requested permissions (camera, microphone, location, calendar, notifications, biometric unlock). Let new permissions of later versions be approved automatically or confirm them again when asked.
3. Sync the app and assign it to a device group as **Required**. Fully managed devices and devices with a work profile are suitable; in the work profile the app's data stays separate from the personal side.
4. Use the Managed Google Play update policy for updates (for example "High priority" for prompt updates). Once all devices are updated, raise the **Minimum client version** under **Policies**.

The QR code and document scanners, the map view and buying Professional need Google Play services on the device. Work phones with a seat of the organisation need no purchase: the seat includes every feature.

Without Google Play: device management systems that only distribute Android apps as files (line-of-business apps) use `FocusFamily-1.4.0-Signed.apk` from the download server. Compare the checksum with `SHA256SUMS`. The APK is signed with Software Notion's key, the Play version with Google Play's

signing key: a device therefore stays with one distribution channel, and switching requires uninstalling (sync the local data or save a backup first). New versions then have to be added to device management manually.

iPhone and iPad through Apple Business Manager

1. In Apple Business Manager, search for "Focus Family" under *Apps and Books* and "buy" the number of licences needed for the location (the app is free).
2. Add the Apps and Books token to device management (Intune: *Tenant administration → Connectors and tokens → Apple VPP tokens*). Focus Family then appears in the app list.
3. Assign the app to a device group as **Required** with **device licensing**; no Apple ID is then needed on the device.
4. Turn on automatic updates for Apps and Books apps and raise the **Minimum client version** after the rollout.
5. Optionally have the managed app and its data removed when the device leaves management.

The app's data is excluded from iCloud and computer backups and from device transfer. A replacement device starts with setup and gets the data through synchronisation with the server. To prevent purchases on work phones, block in-app purchases with a device restriction; devices with a seat need no purchase.

Desktop app

- **Windows:** the MSI installs per machine and supports unattended installation: `msiexec /i FocusFamily-1.4.0-x64.msi /qn /norestart`. Add it to Intune as a *Windows app (Win32)* or *line-of-business app*; the MSI product code serves as the detection rule.
- **macOS:** the PKG is signed with Developer ID and notarised by Apple and installs through Intune (*macOS app (PKG)*), Jamf or `sudo installer -pkg FocusFamily-1.4.0-AppleSilicon.pkg -target /`.
- **Linux:** roll the packages out through your own package distribution or configuration management (`apt install ./FocusFamily-1.4.0-amd64.deb` or `dnf install ./FocusFamily-1.4.0-x86_64.rpm`).

The desktop app finds new versions itself and installs them on request; every update is signed. On Windows the installation needs administrator rights, on Linux the package manager. On computers without local administrator rights, roll new versions out through software distribution instead.

Connecting to the server

After installation each app connects once to the organisation's server: with a **setup QR code** from the admin panel (*Users → Show setup QR*, see the admin panel handbook) or, with single sign-on, with the server address and the organisation account. The device then appears under **Devices** and is assigned to the person's seat.

Recommended device policies

- Enforce a screen lock with passcode or biometrics and device encryption; Focus Family's app lock adds a further layer.
- Keep operating systems at least at the versions in the table above and have security updates installed promptly.

- If a device is lost or stolen, remove it in the admin panel under **Devices** (this ends synchronisation and the licence) and also lock or wipe it through device management (see "Lost or stolen device").

Security

Transport and hardening

- Production is **HTTPS only**; TLS terminates at the proxy.
- **Forwarded headers** are accepted only from `REVERSE_PROXY_NETWORK` and `REVERSE_PROXY_IP`, and with `REVERSE_PROXY_ALLOWED_HOST` only for the configured host name. 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; the security headers also apply to error pages, and every panel action (`/admin/act/*`) requires an antiforgery token.
- **Rate limiting** per IP with fixed one-minute windows: general `RateLimiting:GlobalPermitLimit` (3000), sign-in, account, MFA, passkey and SSO endpoints `AuthPermitLimit` (100), device enrolment `DeviceEnrollPermitLimit` (10), crash reports `DiagnosticsPermitLimit` (20), uploads to the activity record archive `ArchiveUploadPermitLimit` (60). Repeated wrong passwords lock the account temporarily. If a VPN gateway translates all devices to one address (NAT), they share one quota.
- **Account deactivation:** a deactivated account loses access immediately – panel sessions, app tokens and sync are rejected, as are tokens with an outdated security stamp (for example after a password change). Sync is not possible without a device ID.
- **Sign-in without account disclosure:** unknown, locked and deactivated accounts as well as wrong passwords receive the same answer after the same hash check.
- 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.

Case transfer from the apps: `POST /api/v1/cases/{caseId}/transfer` with `{ "userId": …, "keepAccess": true|false }` hands a case over to another practitioner. Only the current owner with write access to the case may do this; the target must be an active practitioner account of the same organisation. Responses: `204` on success, `400` with `{ "code": "transfer.invalid_target" }` for an invalid target, `400` with `{ "code": "transfer.private_keep_access" }` when a private case is to be transferred with `keepAccess: true` (private cases can only be transferred without keeping access; the case remains unchanged; `GET /api/v1/cases/{caseId}/participants` returns `caseIsPrivate` on every entry so the apps disable "Keep access" up front), `403` for non-owners, `404` if the case is not visible to the caller. Every transfer is logged in the audit as "Case transferred". The apps only call the endpoint while online and only

after a complete pre-transfer sync; if that sync fails or pending changes, open conflicts or files not yet uploaded remain, the transfer is cancelled without calling the server.

Permission ceiling: every account change in the panel and in the admin API (permissions, password, MFA, passkeys, deactivation, sign-in method, areas) requires that the target account holds no rights beyond the actor's own scope. Area-bound admins only change accounts whose areas and teams lie entirely within their scope, and never grant *Operations admin* or *Manage areas* (`AreaManage`). The last active local system admin account with a password is protected against deactivation and switching to SSO.

Area mode: when area mode is on (panel → **Areas**), the `AreaScopeEvaluator` limits area-bound practitioner admins – in addition to their permissions – to the practitioners of their areas and those practitioners' cases; filters run in SQL. Every action under `/admin/act/*` passes an endpoint filter that checks the target and rejects fail-closed when no target is declared (redirect `err=not_found`, audit `area.scope_denied`). Switching on and off requires confirming the sign-in again (password plus TOTP, or passkey). Apps, sync protocol, `MeResponse` and policies stay unchanged; with the mode off the evaluator changes no query.

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:

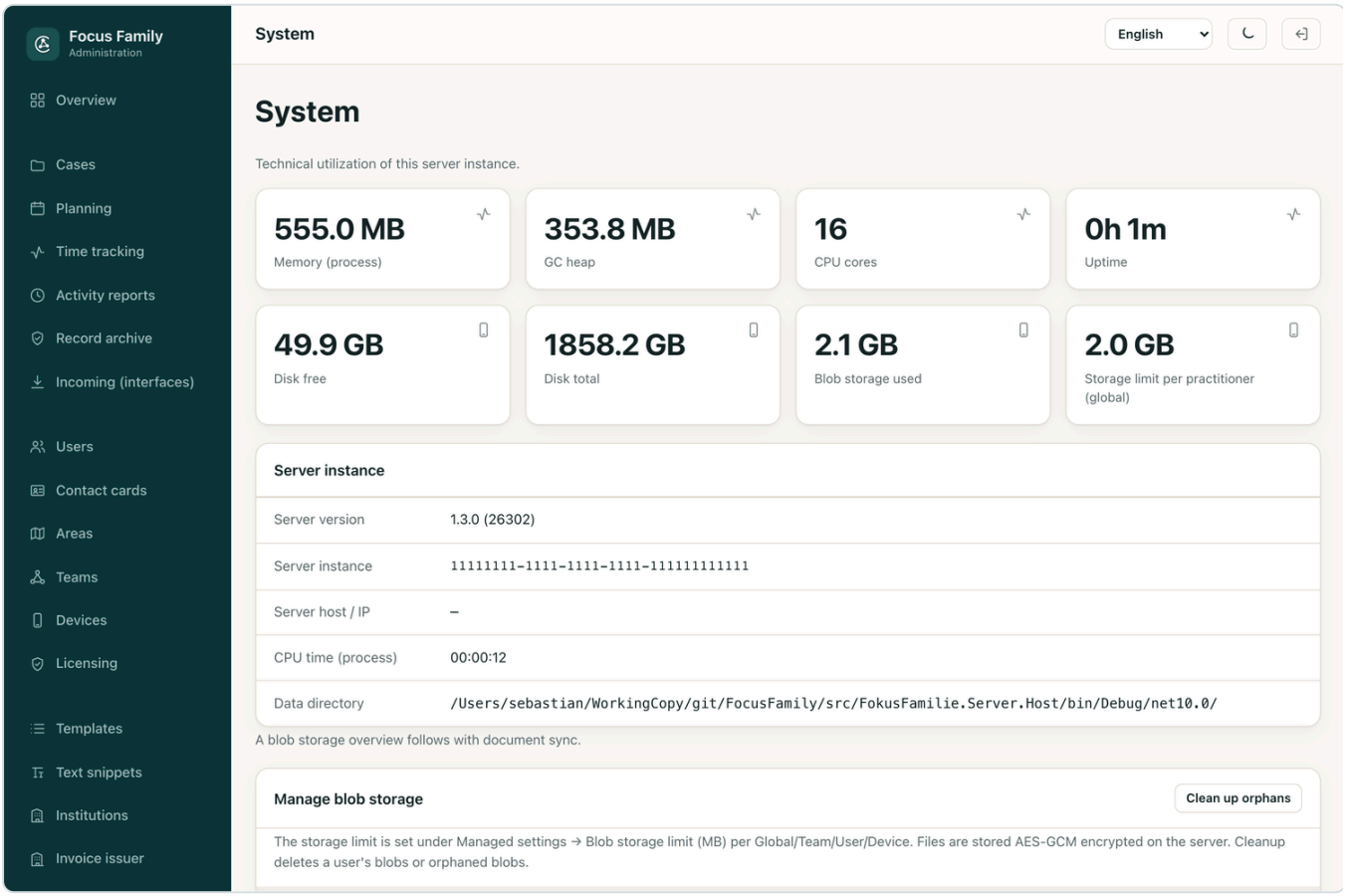
1. Keep the old private key as the signer, add the **new public** key to `OFFLINE_GRANT_ADDITIONAL_PUBLIC_KEY` and raise `OFFLINE_GRANT_KEYSET_VERSION`. Wait until all clients have synced at least once.
2. Switch `OFFLINE_GRANT_KEY` to the new private key, keep the **old public** key for the transition 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.

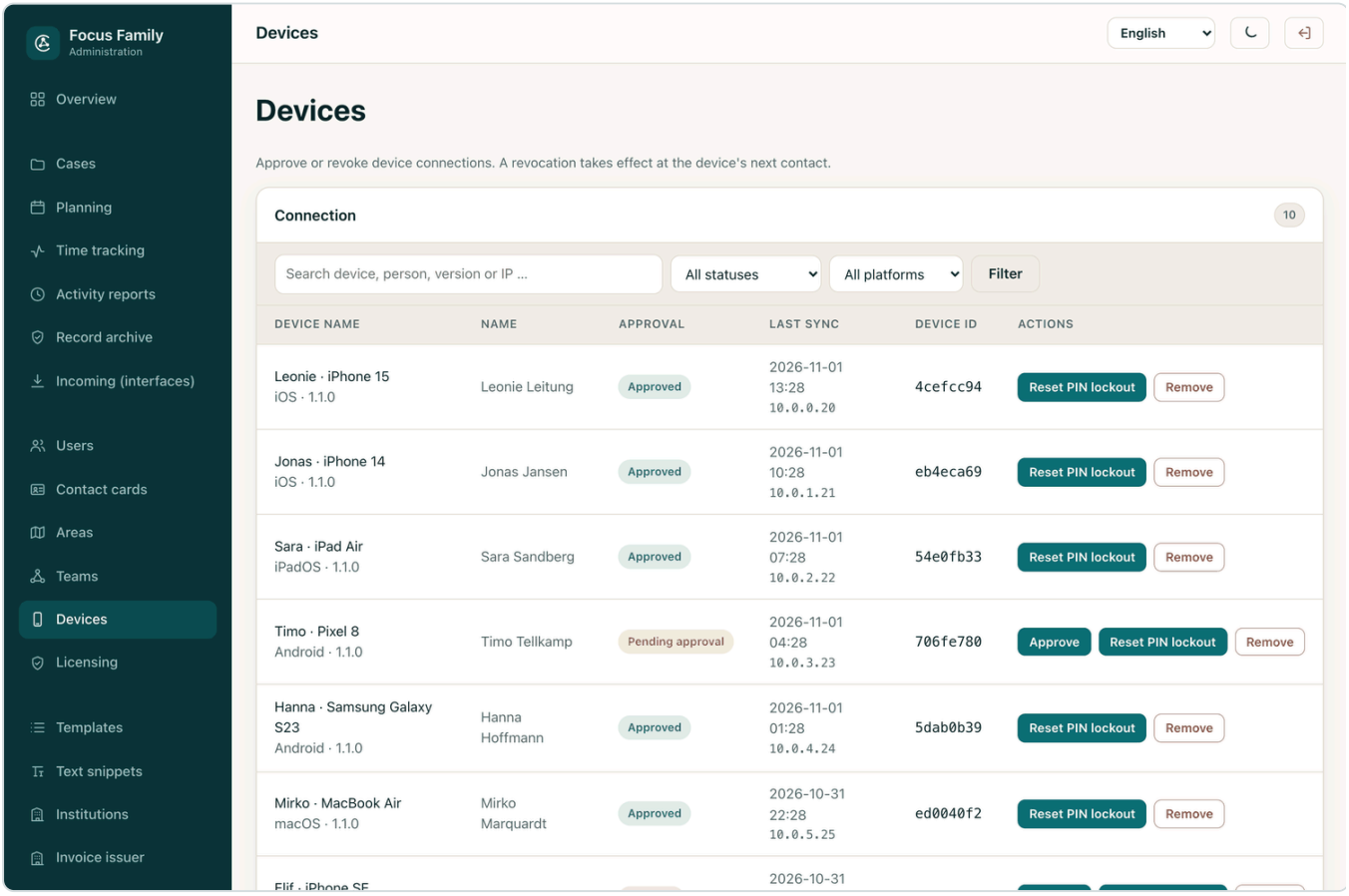
> ⚠ **Cleaning deletes irreversibly.** The function removes a user's uploaded documents and media – or orphaned files – **permanently** from the server. Content that no longer exists on any device is then lost. A confirmation prompt appears first.

The same page carries the card **Key for file attachments** (permission *System configuration*). It names the active key id and how many envelopes are still on an old key. Add a new key to `BlobStorage:EncryptionKeys` **first**, point `BlobStorage:ActiveKeyId` at it, restart, and then run the **rewrapping**. Remove the old key only once the card reports nothing left.

The run replaces a file only while it is unchanged; files edited at the same time keep their newer version and are reported as skipped. An envelope that cannot be read is listed separately and prevents the "complete" result — deliberately so, because its key may still be needed. In an installation with several instances only one run works at a time; a second call reports that instead of writing in parallel. Start and end appear in the audit.

Below it sits the **Monitoring** card (permission *Configure monitoring*). It switches the monitoring interface on, creates and renews the access key, restricts it to particular addresses and decides whether operational counts are included. The card also shows the full addresses ready to copy. How to connect this to Prometheus, Grafana or Uptime Kuma is described under "Operating and monitoring".

Devices

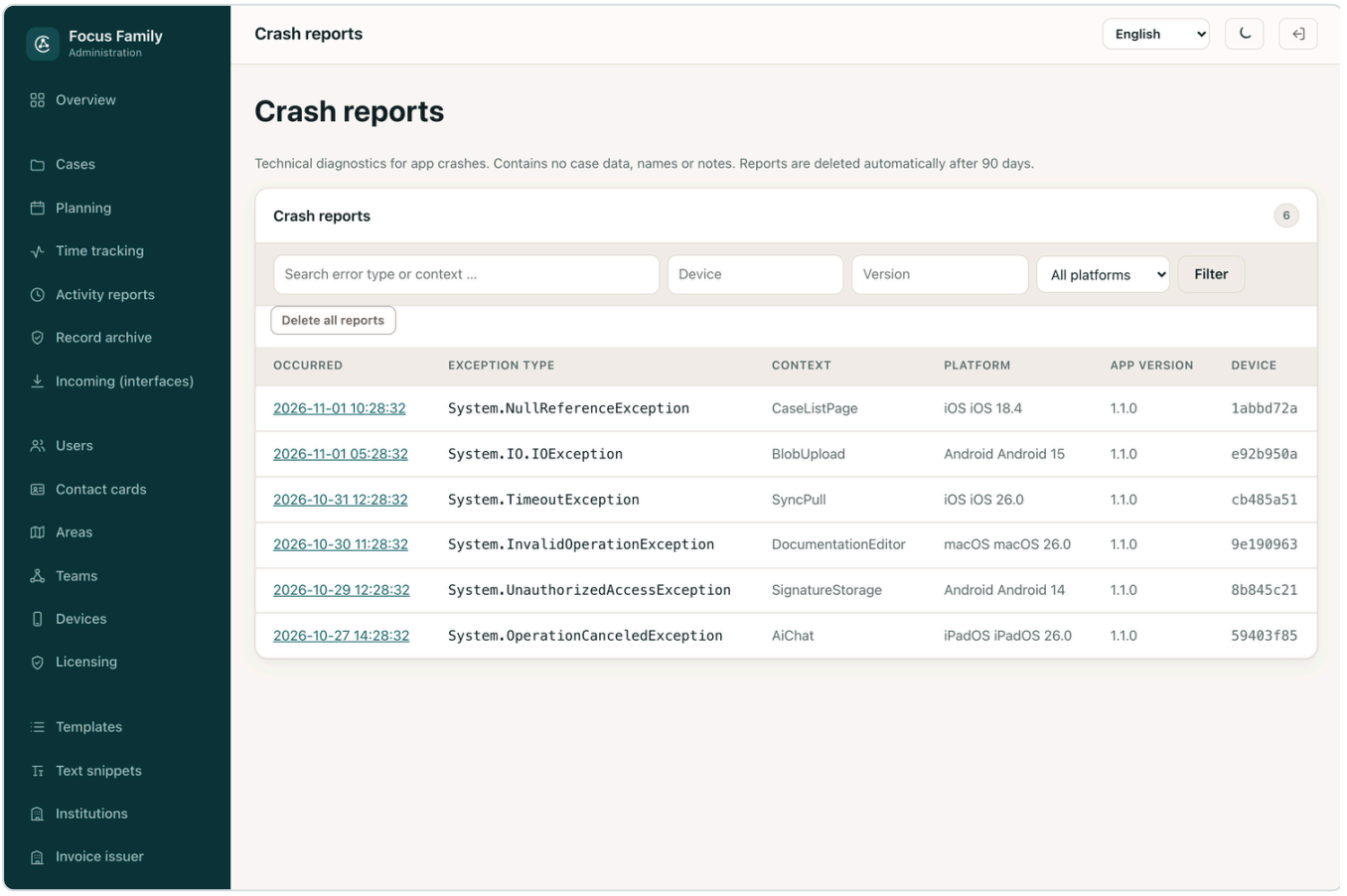


Approve, block, reset the PIN lock, remove a device and issue re-enrolment codes.

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.

The apps send the device model with every request (header `X-Device-Model`, e.g. "iPhone Duo", at most 64 characters, column `Devices.DeviceModel`) and a user-assigned device name (`X-Device-Name`, at most 100 characters, only if it says more than the device type); both are URL-encoded, control characters are removed. The app version is taken from `X-Client-Version` on every request if it is a valid version number and stored as `Major.Minor[.Build]`. *Devices* and the device list under *Licensing* show the model with the name below it.

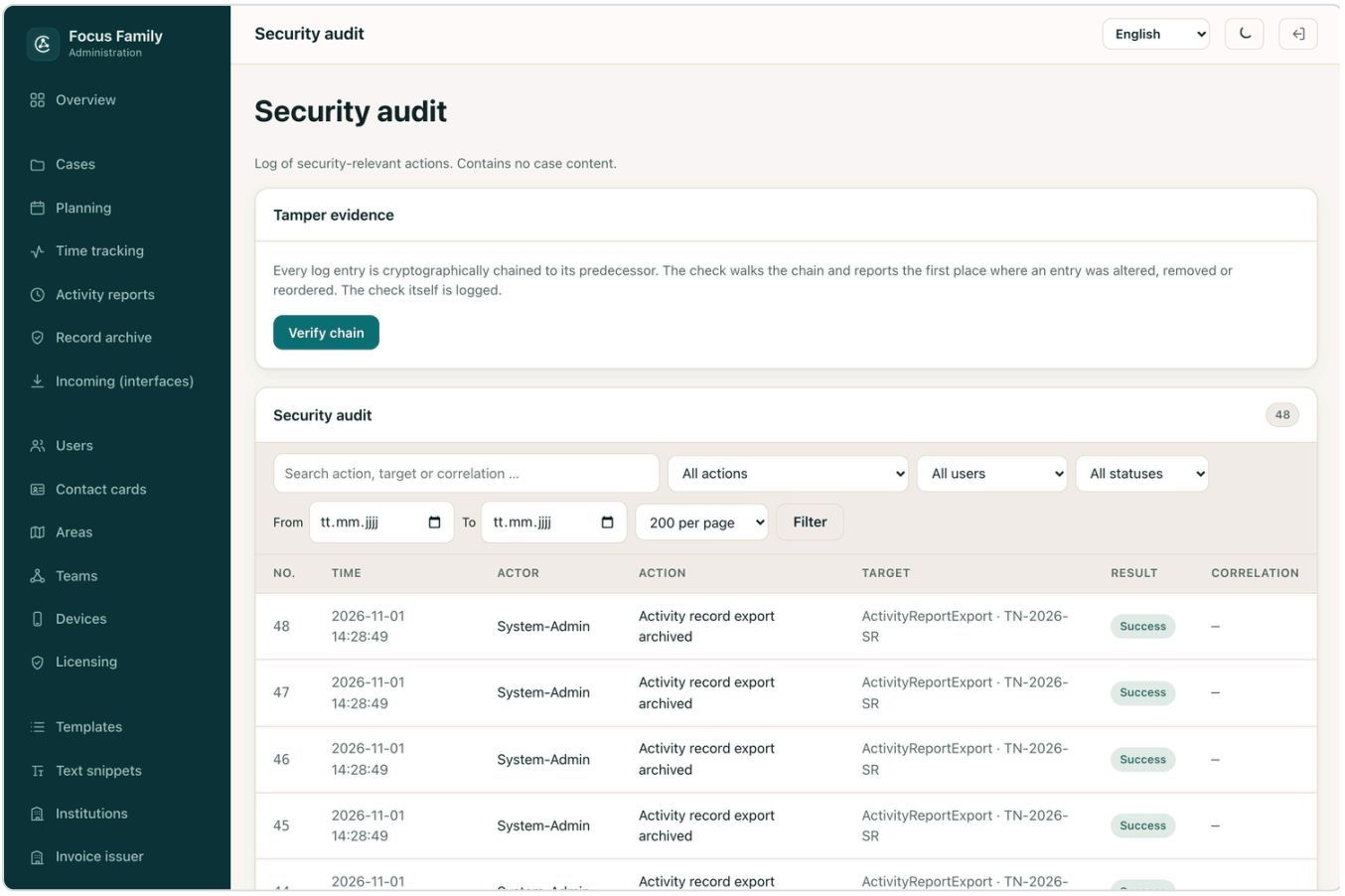
Crash reports



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. The server deletes reports automatically after `Diagnostics:CrashReportRetentionDays` (default 90 days); in the panel single reports or all reports of the organization can be deleted immediately with the permission *Manage crash reports* (audit entries `diagnostics.crash_report_deleted` and `diagnostics.crash_reports_purged`). Intake is limited per sender IP to `RateLimiting:DiagnosticsPermitLimit` reports per minute (default 20) and to 50 reports per device and day; oversized reports are rejected.

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.

Case history (table `CaseTimelineEntries`, schema version 23): Separately from the audit, the server keeps a history of changes per case. It is written in the same database transaction as the change itself – on sync push (`SyncService`, including the device), for panel actions, the app endpoints for sharing, cover arrangements and transfers, the admin API (`assign-team`, `assign-user`, `acl`) and when cover arrangements expire on schedule (source "System"). It stores the time, account, device, source, entity type and ID, action and the names of changed fields from comparing the old and new payload – never values and never names. Technical fields (timestamps, versions, checksums, derived fields) do not count; changes without a business field create no entry; records without a case are not recorded. The application role has no `UPDATE` right on the table. Entries are deleted together with a case when it is permanently deleted after the deletion review; when single records are permanently deleted, only a "permanently deleted" note remains per record. There is no separate retention period – the history lives as long as the case, just like the audit. Person names are resolved from the user account only when displayed.

Activity record archive

App, desktop and panel store exported activity records encrypted in blob storage (table `ActivityReportExports`). Every stored file counts against the practitioner's **blob storage limit**; the permitted file size follows the upload policy, and uploads are limited by `RateLimiting:ArchiveUploadPermitLimit` (default 60 per minute and IP). When an included time entry or

case is permanently deleted, the server removes the file; number, version and checksums remain as evidence.

Licensing

The menu item **Licensing** shows the state of the organisation licence (without a subscription but with sponsored seats: "Sponsored"), the used seats as "assigned / purchased + sponsored", the note on sponsored seats, unlicensed assignments ("Assigned, unlicensed") and the registered devices with platform and app version. 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.

Checking time synchronisation

The server keeps all timestamps in UTC and is authoritative for time. A wrong host clock has immediate effects:

Mechanism	Time reference	Effect of skew
Device proof on every sync and file request	±5 minutes	all devices are rejected
Offline grants	issue time at most 5 minutes in the future	devices discard the grant
Tokens, setup QR, passkey sign-in	5 minutes to 1 hour	sign-in fails or tokens live too long
Two-factor codes (TOTP)	30-second steps	codes are rejected
SSO (OIDC and SAML)	validity windows of responses, sign-in re-confirmation at most 5 minutes old	error `sso` or `sso_reauth`
Licensing server	2 minutes tolerance	licence check and seat assignment fail
Audit log, retention and deadline jobs	timestamps and periods in days	events cannot be correlated, deadlines wrong

Check at commissioning and in routine operational checks:

```
timedatectl status                # "System clock synchronized: yes" and "NTP service: active"
chronyc tracking                  # with chrony: "Leap status : Normal", offset in the
millisecond range
chronyc sources -v                # reachable time sources
docker compose exec app date -u   # same UTC time as on the host
```

If synchronisation is off, `sudo timedatectl set-ntp true` (systemd-timesyncd) or installing chrony with the internal time server as source fixes it. The host clock belongs in the host's monitoring; the admin panel does not show clock offset. On managed end devices, enforcing automatic time via MDM is recommended.

Monitoring

The server ships its own monitoring interface. It sends nothing outwards — your tools fetch the numbers.

Address	Purpose	Access
`/health/live`	the process is alive	open
`/health/ready`	process and database ready	open
`/status`	returns `up` as plain text	open
`/metrics`	metrics in Prometheus format	access key
`/monitoring/status`	the same numbers as JSON, with `status` = `ok` / `degraded`	access key

Setting it up (permission *Configure monitoring*, offered as the second checkbox in the *View system health* row): **System** → **Monitoring** → **Create key**. The key is shown **once**; afterwards only its checksum remains in the database. Then **switch the interface on**. Without a key, `/metrics` and `/monitoring/status` answer `404` — from outside there is no sign they exist. The same applies while the interface is switched off: creating or renewing the key does **not** switch it on. With the interface on and a wrong or missing key the answer is `401`.

Optionally enter the IP addresses or CIDR ranges of your monitoring server under **Allowed addresses**. Behind a reverse proxy the forwarded address counts; that requires `ReverseProxy:Enabled` and your proxy listed under `KnownProxies`. If the address cannot be determined, the scrape is refused.

Prometheus:

```
scrape_configs:
  - job_name: fokusfamilie
    scrape_interval: 60s
    metrics_path: /metrics
    scheme: https
    authorization:
      type: Bearer
      credentials: ffmon_…
    static_configs:
      - targets: ["server.organisation.example"]
```

Uptime Kuma: monitor type *HTTP(s) - Keyword* on `https://…/monitoring/status`, keyword `"status":"ok"`, under *Headers* `{"Authorization": "Bearer ffmon_…"}`. For a plain availability check use `/health/ready` without a key.

Worth alerting on:

Metric	Meaning
`fokusfamilie_ready`	0 = the database cannot be reached **or** a metric group could not be collected
`fokusfamilie_collection_complete`	0 = at least one group is missing; `failedGroups` in the JSON answer names which
`fokusfamilie_disk_used_ratio`	data volume; above 0.85 it gets tight
`fokusfamilie_blob_used_ratio`	share of the attachment quota in use
`fokusfamilie_privacy_requests_overdue`	data-subject requests past their statutory deadline — should always be 0
`fokusfamilie_audit_failures_24h`	rejected security-relevant actions
`fokusfamilie_break_glass_active`	emergency accesses currently open
`fokusfamilie_retention_last_run_age_seconds`	age of the last retention run (change log and deletion markers, daily); well above 90,000 means the run is not happening
`fokusfamilie_blob_key_rotation_unreadable`	attachments whose key is missing
`fokusfamilie_blob_scan_age_seconds`	age of the last attachment sweep; permanently high values mean the sweep no longer completes
`fokusfamilie_audit_chain_checkpoint_age_seconds`	age of the last chain verification; −1 = never verified
`fokusfamilie_devices_pending_approval`	devices waiting for approval

How often you may scrape: as often as you like. The attachment sweep across the data volume is the only expensive part; its result is reused for **five minutes**, and overlapping scrapes do not queue up — they receive the result already at hand. A *failed* sweep is held for 30 seconds only, so a passing disturbance leaves the report quickly while a permanently broken store is still not walked on every scrape. The interface settings themselves are held for **15 seconds**, so a scrape is still answered while the database is briefly unreachable — in that case `/monitoring/status` answers `degraded` with `503` rather than nothing at all. A key change or switching the interface off takes effect immediately, regardless of that cache; an older state cannot come back — neither through another save running at the same time nor through a scrape that read the settings moments earlier.

During a longer database outage that fallback ends after **ten minutes**: the interface then answers `404` again rather than keep working from details whose validity it can no longer check. A revoked key therefore survives a database outage by ten minutes at most — and on the instance where you revoked it, not for a second.

When a metric is missing: if a single group fails (the attachment sweep or the chain verification, say), the interface still serves every other number, names the failed group in `failedGroups`, sets `fokusfamilie_collection_complete` to 0 and reports `degraded`. A partial failure is therefore never presented as "all is well".

The interface serves **numbers and technical readings only** — no names, no case content, no personal data. **Include operational counts** additionally switches off the number of accounts, teams, cases and devices if not even the order of magnitude should leave the house.

A new key invalidates the current one immediately. Switching on and off, key changes and scope changes all appear in the security audit.

Licence server: the same two addresses (`/metrics`, `/monitoring/status`) exist on the licence server. There is no interface there; the key comes from configuration (`Monitoring__ScrapeToken`, optionally `Monitoring__AllowedNetworks`). Without a configured key both addresses are invisible there too. Alert on:

Metric	Meaning
`fflicense_database_up`	0 = the licensing database cannot be reached
`fflicense_signing_key_usable`	0 = the active signing key is missing or switched off; no further licences can be issued
`fflicense_signing_key_active`	number of active keys; **several are normal during a key change** and not a fault
`fflicense_collection_complete`	0 = a metric group is missing (see `failedGroups`)
`fflicense_identity_deletions_manual_required`	> 0 = a licence identity deletion job waits for manual processing (status `degraded`)
`fflicense_identity_deletions_overdue`	> 0 = a deletion job has been open for more than 24 hours (status `degraded`)

Sync retention and full resync

Once a day the server removes change-log entries that no device needs any more (policy values `changeFeedRetentionDays` and `tombstoneRetentionDays` of the governance API, 180 days each by

default; audit entry `retention.enforced`, metric `fokusfamilie_retention_last_run_age_seconds`). The newest state of every existing record is always kept, so new devices can set themselves up at any time.

A device that has not synchronised for longer than the retention period can no longer catch up step by step. The app and the desktop app then rebuild their data automatically from the current server state: pending local changes are sent first, then the complete state is loaded, and records that were deleted on the server in the meantime or are no longer accessible are removed from the device (including attachments). Records that exist only on the device, private records and records with changes not yet sent stay untouched; a changed record that no longer exists on the server appears as a conflict "deleted centrally" (keep locally or discard). No error message appears; if the rebuild is interrupted it simply resumes with the next sync.

Older app versions cannot perform this rebuild. For them the server delivers the complete current state instead, so they keep synchronising. Records deleted or revoked on the server during the long offline period may however remain on such a device until the app is updated to the current version or reinstalled. In the log the server notes devices beyond the retention period ("must perform a full resync").

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. Keep the previous `encryption.pfx` as `deploy/docker/keys/encryption.previous.pfx` and set `OpenIddict__PreviousEncryptionCertificatePath: /keys/encryption.previous.pfx` (password optionally via `OpenIddict__PreviousEncryptionCertificatePassword`, otherwise `CERT_PASSWORD`): the data protection key ring contains keys encrypted with the old certificate and stays readable only this way (licensing installation identity, SSO secrets). Do not delete the old certificate while such keys exist. 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 container exits at start (production)	The start-up check found missing or unreadable settings. `docker compose logs app` lists them under "Focus Family cannot start" with the command that generates each one (e.g. `OFFLINE_GRANT_KEY`, `BLOB_ENCRYPTION_KEY_BASE64`, `deploy/docker/keys/*.pfx` / `CERT_PASSWORD`, `SERVER_INSTANCE_ID`, initial admin on an empty database). Complete `.env` and run `docker compose up -d`
401 or redirect loops behind the proxy	`X-Forwarded-Proto=https` is not forwarded, the proxy is neither in `REVERSE_PROXY_NETWORK` nor at `REVERSE_PROXY_IP`, or `X-Forwarded-Host` does not match `REVERSE_PROXY_ALLOWED_HOST`
All devices share one rate limit	The app does not trust the proxy: `docker network inspect focusfamily_web` must show the subnet from `REVERSE_PROXY_NETWORK` (otherwise recreate the network with `down` / `up`); a proxy on the host needs `REVERSE_PROXY_IP`
`docker compose up` reports "Pool overlaps"	The range from `REVERSE_PROXY_NETWORK` is already used on the host – enter a free `/24`
Panel update ends with `downgrade_refused` or `version_mismatch`	The offered image is not newer than the running one or does not carry the requested version – not an installation fault, check the feed
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`
Sign-in via SSO fails	look up the code in the `error=` parameter of the sign-in page in the table *SSO troubleshooting*
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` generated (whole PEM in double quotes) and backed up
- [ ] Proxy attached to `focusfamily_web` (or `REVERSE_PROXY_IP` set), `REVERSE_PROXY_ALLOWED_HOST` set to the panel name, proxy forwards the forwarded headers
- [ ] Neither `docker-compose.noclamav.yml` nor `docker-compose.local.yml` in use
- [ ] initial admins created at the first start, initial passwords replaced at the first sign-in, `BOOTSTRAP_*` values then removed from `.env`
- [ ] `/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 in place: `/health/ready` for availability, an access key created under *System → Monitoring* and the interface switched on, alerts on `fokusfamilie_ready`, `fokusfamilie_disk_used_ratio` and `fokusfamilie_privacy_requests_overdue`
- [ ] `Monitoring__ScrapeToken` set on the licence server (otherwise its interface stays off)

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`
- [ ] monitoring: key still valid, scrapes arriving, no permanent alerts

---

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`, `docs/server/SSO.md` (developer reference) and `deploy/docker/README.md`.

Focus Family · Technical Handbook · Version 1.4.0 · October 2026