ADMINISTRATION GUIDE V26.07.27.1120 DENSEDEFENSE · FORTEFIDE

ForteFide Administration Guide

Operating ForteFide end to end: install, prepare endpoints, scan, remediate, roll back, collect signed evidence, and tear down — the 6-step engagement model, the auth-tier cascade, and the leave-no-trace doctrine that backs every one of them.

- Scan — free, always on
- Remediate / Prepare / Signed evidence — licensed
- DDWitness cloud evidence — optional add-on

Where to start

INSTALL

Linux: `sudo dpkg -i ForteFide_Setup_26.07.27.1120_linux.deb` · Windows: run `ForteFide_Setup_26.07.27.1120_Windows.exe` as Administrator

OPEN

Dashboard at `http://127.0.0.1:5000` (loopback-only; first launch sets the ForteFideAdmin password); scanning works immediately, no license required

LICENSE — Unlocks Remediate, Prepare, fleet operations, and signed evidence — see §3

CONTENTS

1 · Product Overview & Licensing Tiers

ForteFide is a compliance-readiness scanner and remediation engine for organizations preparing for a CMMC Level 2 assessment under NIST SP 800-171 Rev 2. It runs from a single host (the scanner) and reaches Windows and Linux endpoints over SSH and WinRM, technically evaluating each against **79 of the 110 controls** across 14 NIST 800-171 r2 families — and, once licensed, remediating findings the customer authorizes.

ForteFide is built around a **leave-no-trace doctrine**. Every state change made on a target — a touched config file, an injected sudoers entry, a created service account, a deployed certificate — is recorded in a per-target reversion bundle. After the engagement, Teardown undoes those changes against that record. What the C3PAO assessor receives is a signed evidence package; what the customer's machine ends up with is its pre-ForteFide state.

> **Tip —** Current build: **v26.07.27.1120**. Coverage figures throughout this guide are sourced live from the scanner-matrix audit and the remediation catalogue — 79 automated + 31 manual = 110 total controls; the remediation catalogue auto-fixes 67 on Linux and 66 on Windows (82 distinct controls across the union of both).

How ForteFide is shaped

- **Scanner host** — the operator's machine. Runs the dashboard on `localhost:5000` and originates every scan, remediation, and rollback action.
- **Targets** — Windows and Linux endpoints reached over the wire. Targets are NOT configured before scanning; ForteFide assesses what is present and reports findings against it.
- **Storage** — encrypted SQLite, machine-bound license cert, JSONL journal, per-target reversion bundles, and a flat audit log — all under `DATA_DIR` (Linux: `/var/lib/fortefide`; Windows: `%ProgramData%\ForteFide`).
- **License** — a single `license.key` file activates remediation. Scanning is always free. Remediation, fleet operations, signed evidence, and prepare are gated.

Dashboard access & authentication

The dashboard binds to `127.0.0.1:5000` (loopback) by default — not reachable from the network on a stock install. Binding to another interface is an explicit opt-in (`FORTEFIDE_BIND_HOST` or `BIND_HOST` in the config); if you do, keep it behind an SSH tunnel rather than exposing port 5000.

Access is protected by a single built-in account, **ForteFideAdmin** — the operator sets only a password (there is no username to choose):

- **First run** — the first visit lands on a one-time setup page to set the ForteFideAdmin password (minimum 12 characters), stored hashed (scrypt) at `DATA_DIR/dashboard_credentials.json`, mode `0600`. No default password ever exists.
- **Sign-in** — the ForteFideAdmin password at `/login`; the session is remembered 30 days per machine (`HttpOnly`, `SameSite=Lax` cookie — which also blocks CSRF against `localhost:5000`).
- **Remote** — tunnel first (`ssh -L 5000:localhost:5000 <scanner-host>`), then sign in with the same password.
- **Automation / API** — `/api/*` also accepts an internal token as an `X-Auth-Token` header (retrieve on the scanner host with `fortefide --show-token`). For scripts and machine-to-machine use only; people sign in as ForteFideAdmin with the password.

Tip — No online password reset by design. If the password is lost, delete `DATA_DIR/dashboard_credentials.json` on the scanner host and restart ForteFide — the next launch returns to the first-run setup page.

Licensing tiers

ForteFide ships in a free scanner tier and a licensed tier. Licensing is priced **per prepared endpoint**, graduated by volume — not a fixed named-plan seat cap.

TIER	ENDPOINTS	SCAN	REMEDIATE	SIGNED EVIDENCE	TYPICAL CUSTOMER
Free / Scanner	Unlimited	Yes	No	Draft only	Pre-purchase evaluation; one-shot compliance temperature check.
Licensed	Per prepared endpoint (volume-banded)	Yes	Yes	Yes	Any CUI contractor — priced from **$89**/endpoint/mo down to **$34**/endpoint/mo at scale (see the License Guide for the full bands).

Note — Seat = distinct prepared endpoint currently held in `_svc_accounts`. Scanning a host does not consume a seat. Preparing DOES consume a seat the moment it succeeds — `/api/prepare-endpoint` registers the target immediately, and that map is what the remediate endpoints count against `max_endpoints`. Tearing down a target frees the seat. A Starter (25-seat) admin preparing 25 hosts cannot prepare a 26th until one is torn down — even if none of the 25 have been remediated yet.

2 · System Requirements

Scanner host

Linux (recommended for fleet operations):

- Debian 11+ / Ubuntu 22.04+ / RHEL 8+ / SUSE 15+ (x86_64)
- libc6 >= 2.28 (standard on all listed releases)
- 4 GB RAM minimum (8 GB recommended for fleets > 50 hosts)
- ~400 MB disk for the customer .deb (Layer 1 airgap packages bundled by default; tooling-only `--no-airgap` builds are ~50 MB but are not the customer-shipping default)
- Root or sudo for installation; the service runs as root by default (privileged ports + `/var/lib/fortefide` writes)
- No Python runtime required — ELF binary compiled with Nuitka 4.0.7; no internet access required at runtime

Windows:

- Windows 10/11 or Windows Server 2019+ (x86_64)
- Administrator privileges for installation
- 4 GB RAM minimum, 500 MB disk
- Windows Firewall rule auto-created for the dashboard port

Target endpoints

PLATFORM	MINIMUM OS	REQUIRED SERVICE	DEFAULT PORT	NOTES
Windows	Windows 10 / Server 2016	OpenSSH Server (sshd) + WinRM	22 / 5985	SSH preferred over WinRM for both auth and remote PowerShell; WinRM still used for Phase 0 bootstrap on legacy hosts.
Linux	Any glibc-based distro	OpenSSH Server (sshd)	22	Validated: Debian 11/12, Ubuntu 20/22/24, RHEL 7/8/9, Rocky/Alma, SLES 12/15/16, Arch (rolling) — see §19 for per-distro caveats.

Network requirements

- Scanner → target: TCP 22 (SSH), TCP 5985 (WinRM HTTP)
- No outbound internet required by default — the scanner does not phone home; license verification is offline (the license.key file is cryptographically bound to this specific machine). An opt-in periodic

license-status check can be enabled in Settings — see §3.

- Operator → scanner: HTTP 5000 on localhost (or a custom port selected at install time)

3 · Installation & License Activation

Linux (.deb)

ForteFide on Linux ships as a Debian package containing a Nuitka-compiled ELF binary — no Python interpreter dependency on the target host.

```
# Customer .deb (Layer 1 packages tree bundled)
sudo dpkg -i ForteFide_Setup_26.07.27.1120_linux.deb
```

What the installer creates:

- `/opt/fortefide/` — ELF binary, templates, static assets, vendored packages tree (Layer 1 bundles, default-on)
- `/var/lib/fortefide/` — `DATA_DIR`. Holds license.key, scan.db, journal/, certs/, target_profiles.json, `_svc_accounts` persistence
- `/var/log/fortefide/fortefide.log` — audit log (append-only; INFO by default, DEBUG with `--debug`)
- `/etc/systemd/system/fortefide.service` — systemd unit; enabled and started by post-install
- Pre-EULA acceptance is read from `/var/lib/fortefide/.eula_accepted` — if absent, the dashboard shows the EULA modal on first load

Windows (.exe)

Windows installs a Nuitka-compiled binary. The dashboard runs as a foreground process — it is NOT installed as a Windows service. After install, the dashboard launches in the operator's browser automatically.

- Right-click `ForteFide_Setup_26.07.27.1120_Windows.exe` and choose Run as Administrator
- Default install path: `C:\Program Files\ForteFide\`
- `DATA_DIR` on Windows: `%ProgramData%\ForteFide`
- Log path: `%ProgramData%\ForteFide\logs\fortefide.log`
- Start Menu entry, Add/Remove Programs registration, and a Windows Firewall rule for the dashboard port are all created by the installer

First launch

On first run the dashboard shows the EULA (unless the marker file is pre-placed), then the License screen. ForteFide is fully functional in scan mode at this point — a license is not required to scan. The license unlocks remediation, prepare, fleet operations, and signed evidence.

License activation — three paths

All three end at the same place (license.key written to `DATA_DIR`, Pro features unlocked) — they differ in how the key arrives.

- **Path A — Offline (.key file).** Operator generates a request code on the scanner, purchases on densedefense.com, downloads license.key, drag-drops it onto the License popup. Works in fully air-gapped environments.
- **Path B — Online claim link.** Operator clicks "Buy ForteFide" or visits the site directly; Stripe checkout completes and redirects back with `?claim=<token>`. The dashboard auto-calls `POST /api/claim-license` with the token + machine fingerprint. Total clicks from "Buy" to activated: three.
- **Path C — Activation code (no file).** Customer receives a 16-character activation code via email and pastes it into the License popup. Same `/api/claim-license` backend; the code is the token.

> **Tip — Graceful fallback if license-api is unreachable.** `POST /api/claim-license` returns HTTP 503 with a structured "upstream unavailable" message when license-api is not reachable. The popup surfaces this verbatim and offers the offline file-drop path (Path A) as a manual workaround.

Offline activation flow (Path A, detail)

STEP	WHERE	WHAT HAPPENS
1. Open License panel	Dashboard → License button	ForteFide displays the request code in `FIDE-XXXX-XXXX-XXXX` format, derived from a stable machine fingerprint plus the install GUID.
2. Purchase a license	densedefense.com/account	Operator pastes the request code, picks a tier, and pays via Stripe. The license server issues a license.key bound to that request code.
3. Download license.key	Customer email or account portal	A small JSON envelope signed with the DenseDefense Ed25519 private key; tier, expiry, seat limit, and a per-machine AEAD-wrapped key envelope are encoded in the payload.
4. Import the license	Dashboard drag-drop, or `POST /api/import-license`	ForteFide verifies the Ed25519 signature and unwraps the per-machine envelope (fails if the file is used on a different machine).
5. Confirm activation	License panel, or `GET /api/license-status`	Tier, expiration, days remaining, and seat usage display. Expired licenses downgrade to free tier.

Note — License binding & portability. The license is cryptographically bound to the scanner's machine fingerprint, not to your AD domain or network — the binding is enforced by the math (the per-machine AEAD envelope fails to decrypt on a different machine). Moving the scanner host requires a new license bound to the new fingerprint. Always back up `/var/lib/fortefide/license.key` before reinstalling — a `dpkg --purge` wipes `DATA_DIR` including the key.

Optional: online license status check

ForteFide is offline by default. If your environment allows outbound HTTPS, you can opt in to a periodic background license-status check in Settings — informational only; scans, remediation, and evidence generation are never gated on it. Off by default for airgap and CMMC-classified-environment safety.

Upgrade

On Linux, upgrade with `dpkg -i` (NOT `--purge` then `-i`):

```
# Backup the license first
sudo cp /var/lib/fortefide/license.key ~/license.key.bak

# Upgrade (preserves DATA_DIR, journal, license.key)
sudo dpkg -i ForteFide_Setup_26.07.27.1120_linux.deb
```

- `dpkg -i` is an upgrade — it preserves `DATA_DIR`, the journal, license.key, target_profiles.json, and prepared service accounts
- `dpkg --purge` wipes `/var/lib/fortefide` entirely — only use for a clean reinstall, and always back up the license first
- Windows: run the new installer; it upgrades in place and preserves `DATA_DIR`

4 · The 6-Step Customer Journey

ForteFide's operating model is a six-step engagement. Each step produces an artifact (recon list, prepared targets, scan record, remediation results, evidence package, post-engagement state) that the next step depends on. Steps are explicit — the operator advances each one from the dashboard.

#	STEP	WHAT YOU DO	WHAT FORTEFIDE DOES	ARTIFACT
1	Recon	Enter a CIDR range or paste a host list. Click Discover.	Concurrent TCP probes for SSH/WinRM; OS guess from banner; builds a reachable-hosts list.	Discovery record; selectable host list.
2	Prep	Select hosts, enter admin credentials, choose auth method (cert default), click Prepare.	Phase 0 grants NOPASSWD sudo; Phase 0+ ensures baseline; Phase 1 creates fortefide-svc; Phase 2 deploys SSH key + 30-day cert.	Prepared target with cert + svc account.
3	Scan	Select hosts; pick full / quick profile; click Scan.	Authenticates per cascade (Tier 1 → 2 → 3). Runs the 79 technically-scanned controls or the quick subset.	Scan record; per-control MET / NOT MET; compliance percentage.
4	Remediate	Pick controls to remediate — single host, batch, or fleet.	Validates prepared state, walks ordered remediation graph, writes a journal entry per success.	Remediation job record; updated journal + history.
5	Evidence	Click Generate Evidence Package; pick a scan_id; optionally Sign.	Compiles scan results, control evidence, host inventory, and remediation history into a ZIP; signed mode adds an Ed25519 signature and machine fingerprint.	ZIP + PDF + signed .sig file.
6	Out (Teardown / Rollback)	Click Teardown, or Rollback to undo specific remediations.	Reads the per-target reversion bundle, undoes config changes in reverse order, removes fortefide-svc, cert/key, injected sudoers, CA trust file.	Target restored to pre-ForteFide state; signed evidence remains the artifact of record.

Tip — There is no "wizard" that drives all six steps automatically — each step is an explicit operator action with a stop point. A compliance assessment is a deliberate engagement, not a black-box automation.

5 · Authentication Tier Hierarchy

ForteFide reaches every prepared target through one of three auth mechanisms. Higher tier is preferred; falling back to a lower tier when the higher one is unavailable is by design, not a failure. The Tier 1/2/3 *labeling* is uniform across platforms, but the underlying *transports* differ between Linux and Windows.

Tip — Tier 1 cert auth is confirmed on every supported Windows server version, including Windows Server 2012 R2. The log signature to look for is `[PREPARE] <ip>: cert public key added to administrators_authorized_keys` — when present, `/api/svc-accounts` reports `auth_method == "certificate"` for that host within seconds.

TIER	MECHANISM	LIFETIME	ROLE	WHEN IT FIRES
1 (preferred)	SSH certificate signed by license-derived CA; 30-day validity per target	30 days; auto-renewed on next prepare	Steady-state, strongest — license is the root of trust.	Always tried first when a valid cert is deployed.
2	Long-lived Ed25519 SSH key (scan_key), deployed during prepare	Until rotated by operator	Steady-state fallback.	Tries when Tier 1 fails but the key is still in authorized_keys.
3	Password (svc account's randomized 20-char password, or operator-supplied admin/root password)	Per-engagement	Bootstrap mechanism only — inserts the cert + key onto the target during Phase 1/2.	Phase 0/0+/1/2 of prepare; rare emergency path if Tier 1 + 2 both fail post-prepare.

OS-specific transports (what "Tier 1" actually means)

TIER	LINUX TRANSPORT	WINDOWS TRANSPORT
1	SSH client certificate (signed by FF's license-derived CA) over OpenSSH, port 22	SSH key over OpenSSH-on-Windows + PowerShell, port 22 — the preferred surface, since the SSH cert path is fragile on Windows
2	Long-lived Ed25519 SSH key in authorized_keys, OpenSSH port 22	WinRM certificate auth, HTTPS port 5986 — fallback when OpenSSH-on-Windows isn't reachable
3	Password, OpenSSH port 22	WinRM NTLM password, HTTPS port 5986 (HTTP 5985 only if allowed) — bootstrap-only during Prep

Windows transport library: SSH + PSRP + WinRM

TRANSPORT	PORT	LIBRARY	USE CASE
`auto`	22/5985	(probe + select)	Default — probes 22 then 5985; cascades SSH > PSRP > WinRM.
`ssh`	22	paramiko	Windows OpenSSH server reachable; PowerShell via `-EncodedCommand`.
`psrp`	5985	pypsrp	PowerShell Remoting Protocol — used when SSH is unreachable; supports cert + NTLM auth.
`winrm`	5985	pywinrm	Bare WinRM with NTLM password — last-resort fallback; ~8 KB encoded-command limit.

Override per-call by including `"transport": "ssh"` (or `psrp`, `winrm`) in the `/api/prepare-endpoint` or `/api/teardown` JSON body. Windows Server 2012 R2's OpenSSH backport does not accept Ed25519 host CAs the same way modern builds do; ForteFide automatically falls back to PSRP/WinRM for those targets.

Where the cert chain comes from

The license.key is the root of trust. ForteFide HKDF-derives a private half from the license payload, combines it with a binary-side half, and produces an SSH CA keypair scoped to the scanner. That CA signs a 30-day Ed25519 certificate per prepared target. The CA private key is wiped from scanner memory after each prepare run; the CA public key is written to `/etc/ssh/fortefide_ca.pub` on the target so sshd can validate certs against `TrustedUserCAKeys`.

Tip — After prepare, `/api/svc-accounts` shows the realized auth method per host — certificate, ssh_key, or password — and it tracks reality: if cert deployment failed silently, `auth_method` downgrades rather than claiming the higher tier.

5a · Jump Host Recon — Segmented Production Networks

Many CMMC contractors run their production fleet on a segmented network behind a bastion / jump host. The scanner cannot reach the targets directly — the segment is reachable only by SSH-tunneling through the bastion. ForteFide's Jump Host recon path is built for exactly this topology: the operator supplies bastion SSH credentials and a remote CIDR (or remote AD config); ForteFide authenticates to the bastion, opens an SSH tunnel, and probes the remote segment through it. For **Linux** targets, every subsequent operation — prepare, scan, remediate, evidence, teardown — automatically re-uses the same tunnel path.

> **Note — Windows targets over a bastion are not supported this release.** The WinRM transport is not yet wired into the jump-host tunnel, so a Windows target reachable only through the bastion cannot be scanned or remediated over it. Reach Windows CUI endpoints from a scanner with direct WinRM connectivity (Network or AD recon).

When to use Jump Host mode (vs AD or Network)

RECON MODE	USE WHEN	TRADE-OFFS
Active Directory	Targets are domain-joined and the scanner can reach the DC directly.	Fastest, most accurate scope; Linux non-joined hosts are not enumerated.
Network Scan	Targets share an L3-routable subnet with the scanner.	Discovers unmanaged endpoints; noisy on the wire.
Jump Host	Targets are isolated behind a bastion / DMZ / compliance-tightened segment.	One additional SSH hop; bastion creds are operator-supplied each session (not persisted); Linux ops re-use the tunnel automatically. Windows-over-bastion is not supported this release.

What the operator supplies

- Bastion address + SSH port (default 22)
- Bastion SSH username + password — held only in operator memory + the active recon session; never written to disk or `/api/svc-accounts`
- Target method: Network (CIDR sweep through the tunnel) or AD (LDAP query against a DC on the remote segment)
- Remote CIDR, or remote DC IP + domain + DC credentials

Discovery returns the same host objects as Network or AD recon, with one extra field: each Linux host's record carries the bastion path, so Prepare/Scan/Remediate/Teardown auto-tunnel without re-prompting

for bastion credentials. (Windows targets over the bastion are not supported this release — see the note above.)

Failure modes

- Bastion SSH auth failure — recon halts cleanly; no tunnel opened; no state recorded.
- Bastion drops mid-engagement — FF logs the failure per host; prepared certs remain on targets; the next session re-establishes the tunnel and resumes via Tier 1 cert auth.
- Target unreachable from the bastion — marked unreachable in scan results; other targets continue normally.
- Operator doesn't need bastion creds at next-session resume once the first Prepare has succeeded — the cert/key cascade carries the rest.

Security properties

- Bastion credentials are never written to `/api/svc-accounts`, `DATA_DIR`, the journal, or any evidence package.
- All scanner-to-target traffic transits the customer's bastion; bastion logs capture the full engagement footprint.
- Per-target reversion bundles and the journal still record state changes — tunneling does not affect leave-no-trace or evidence integrity.
- If the bastion itself is in CMMC scope, prepare it like any other target.

6 · Endpoint Preparation (Phase 0 / 0+ / 1 / 2)

Prepare is the operation that takes a recon-ed target and turns it into something ForteFide can reach with strong auth. The endpoint moves through four phases on each prepare run.

Phase 0 — Grant NOPASSWD sudo (Linux only)

ForteFide Prepare requires that the bootstrap user can run sudo without a password. Phase 0 verifies this and, if missing, writes `/etc/sudoers.d/<bootstrap_user>-nopasswd` using the password the operator just typed. This is the load-bearing requirement for the rest of prepare.

- Debian/Ubuntu cloud-init users ship with NOPASSWD by default; Phase 0 is a no-op
- RHEL/Rocky/Alma cloud-init also typically NOPASSWD; if not, Phase 0 writes the override
- SLES has `Defaults targetpw` (sudo expects root's password) — Phase 0 writes `Defaults:<user> !targetpw` to neutralize it
- Arch (lab convention): bootstrap user has NOPASSWD via `/etc/sudoers.d/10-scanuser`; Phase 0 is a no-op

Phase 0+ — Baseline ensure

Ensures the target has openssh-server enabled, sudo present, and openssl present. On cloud-init VMs this is a probe-only no-op. On a barebones target, Phase 0+ uses the airgap fallback chain (Layer 3 → Layer 2 → Layer 1) to install what's missing — only after the operator has opted into airgap mode (see §13).

Phase 1 — Service account creation

ForteFide creates a dedicated service account named `fortefide-svc` on the target — the only account ForteFide uses post-prepare; the operator's admin credentials are not stored or reused outside Phase 0/0+/1/2.

- Linux: `useradd -m fortefide-svc`, 20-char random password, `/etc/sudoers.d/fortefide-svc` with NOPASSWD ALL
- Windows: `New-LocalUser fortefide-svc`, added to Administrators + Remote Management Users. W2012R2 fallback: `net user` + `net localgroup`
- Recorded in `DATA_DIR/_svc_accounts.json`; password is wiped from scanner memory after Phase 2 — only the cert/key remains as the steady-state credential

Phase 2 — Cert + key deployment

Derives the SSH CA from the license, issues a 30-day cert for fortefide-svc, deploys the public key + cert to the target, and (on Linux) installs the CA root in sshd.

- Generate `scan_key` (Ed25519) on the scanner if not already present
- Push public key to authorized_keys — Tier 2 is now live
- Derive SSH CA from license HKDF; issue 30-day cert via `ssh-keygen -s`; cert principal = `fortefide-svc`
- Push cert.pub to authorized_keys — Tier 1 is now live
- Linux: write `/etc/ssh/fortefide_ca.pub` and append `TrustedUserCAKeys` to sshd_config; restart sshd
- CA private key is wiped from scanner memory; only the public key persists

Per-target cert directory layout

```
DATA_DIR/certs/<target_ip>/
  key         -- private key for the cert (paramiko presents this)
  key.pub     -- public key matching key
  cert.pub    -- 30-day SSH certificate, signed by CA, principal=fortefide-svc
  metadata.json -- issued_at, expires_at, principal, ca_fingerprint
```

Every artifact prepare creates is also recorded in the per-target reversion bundle so Teardown can undo precisely what Prepare did: the fortefide-svc user, its sudoers entry / Administrators membership, the authorized_keys append (scan_key.pub + cert.pub), the CA trust file (Linux only), the sshd_config `TrustedUserCAKeys` append, and the watchdog cron + script (see §11).

Tip — What Prepare does NOT touch: it does not modify firewall rules, install antivirus, modify password policy, change AD/domain settings, install agents, or run any persistent service on the target. The watchdog is a single cron entry with a heartbeat script — not a daemon.

DC detection and the small-contractor model

ForteFide treats Windows domain controllers as first-class scan targets, not out-of-scope infrastructure. This reflects the typical small-CMMC-contractor reality: a 25-employee defense contractor pursuing CMMC L2 has one server doing AD/file-share/print (their de-facto DC and everything-server) plus 20-24 endpoints. There is no separate "infrastructure tier" out of CMMC scope — the DC is in scope, and the customer is both the AD policy authority and the entity scanning their own AD.

DEFAULT BEHAVIOR

- Prepare, Scan, and Remediation all run against the DC like any other target.

- `DC_MANUAL_CONTROLS` (IA.L2-3.5.6, CM.L2-3.4.6) are DC-aware: they APPLY on a DC (the customer IS the policy authority for their own AD) and SKIP on a domain member (AD overrides local anyway).
- Teardown's multi-select dialog default-checks the DC the same as any other prepared host.

HOW FORTEFIDE DETECTS A DC

During prepare, ForteFide queries the target via WinRM:

```
(Get-CimInstance Win32_ComputerSystem).DomainRole
```

`DomainRole` values: 0 = Standalone Workstation, 1 = Member Workstation, 2 = Standalone Server, 3 = Member Server, 4 = Backup Domain Controller, 5 = Primary Domain Controller. ForteFide records `is_dc = (DomainRole >= 4)` on the svc account record; the flag is cached per-target and cleared on teardown.

LARGER ORGS WITH A SEPARATE-DC ARCHITECTURE

- Mark the DC's svc-account record with `tearable: false` to exclude it from the multi-select Teardown dialog.
- Exclude the DC IP from the prepare scope — prepare is always operator-initiated against an explicit IP.
- If accidentally prepared, run `/api/teardown-endpoint` against just that one host immediately.

Teardown UX — multi-select dialog

ForteFide replaces the single-action Teardown button with a multi-select modal. Auto-teardown stays disabled; the operator decides which prepared hosts come down and when.

CLICK PATH

- Click Teardown in the dashboard topbar; a modal opens with a checkbox list of every currently-prepared host, each row showing target IP, OS family, realized auth tier, a DC badge if applicable, and a "tearable: false" warning chip if marked non-tearable.
- Default selection: all checked, including DCs (small-contractor model). Hosts marked `tearable: false` default to unchecked with the warning chip.
- Select All / Select None / Invert quick actions; "Teardown Selected (N)" fires `POST /api/teardown-batch` with the selected target list and `auto_rollback = true`.

PER-TARGET PROGRESS + SUMMARY

`/api/teardown-batch` dispatches in parallel (ThreadPoolExecutor, max_workers=4) so a slow Windows host does not stall the rest, with per-target failure isolation. Each row updates its status badge live; a summary line reads "N requested, X succeeded, Y failed."

```
POST /api/teardown-batch
{
  "targets": ["10.0.1.50", "10.0.1.51", "..."],
```

```
  "auto_rollback": true   // optional; default true
}

Response: {results: [...], summary: {requested, succeeded, failed}}
Errors: bad_targets, bad_target, too_many_targets (>256), module_missing (license gate)
```

7 · Scan Operations

Scans are read-only assessments. They never mutate the target. ForteFide technically scans **79 of the 110 controls** across 14 NIST 800-171 Rev 2 families using SSH on Linux and SSH+WinRM on Windows. The remaining **31** are organizational (process/personnel/physical), assessed by attestation rather than a technical probe.

Control coverage by family

FAMILY	NAME	CONTROLS	AUTO-CHECKABLE / MANUAL ON THIS PRODUCT
AC	Access Control	22	Auto: 22. Manual: 0.
AT	Awareness & Training	3	All manual (policy).
AU	Audit & Accountability	9	Auto: 9.
CA	Assessment & Authorization	4	Auto: 2. Manual: 2.
CM	Configuration Management	9	Auto: 9.
IA	Identification & Authentication	11	Auto: 11.
IR	Incident Response	3	Auto: 2. Manual: 1.
MA	Maintenance	6	Auto: 5. Manual: 1.
MP	Media Protection	9	Auto: 8. Manual: 1.
PE	Physical & Environmental	6	Manual.
PS	Personnel Security	2	Manual.
RA	Risk Assessment	3	Auto: 3.
SC	System & Communications Protection	16	Auto: 15. Manual: 1.
SI	System & Information Integrity	7	Auto: 7.

Full scan vs quick scan

MODE	CONTROLS RUN	TYPICAL DURATION	WHEN TO USE
Full	All 79 technically-scanned controls	3-8 min/host (Linux); 5-12 min/host (Windows)	Initial baseline; pre-evidence final pass; periodic audit; before C3PAO assessment.
Quick	Subset of automation-heavy controls; skips the slow filesystem-tree scans	60-90 seconds per host	Post-remediation verification; routine drift checks; CI/CD gating.

Scan throttle & live per-host progress

The scan-launch panel exposes a Scan Threads slider (range 4-16, default 8, remembered in the browser) — how many target hosts ForteFide scans at once. While a scan runs, a live per-host list shows one row per target: a state badge (queued / scanning / done / error), a mini progress bar, and the control currently being evaluated. An unreachable host is marked "error" and counted as resolved, so one dead host never pins the overall bar below 100%.

SCANNER HOST	NETWORK TO TARGETS	HOSTS IN SCOPE	SUGGESTED THREADS
2 cores / 4 GB (minimum)	LAN	Any	4 (floor)
4 cores / 8 GB (typical)	LAN, <5 ms RTT	8-24	8 (default)
8+ cores / 16+ GB	LAN, <5 ms RTT	16+	12-16
Any	VPN / WAN / high-latency (>40 ms RTT)	Any	4-6
Any	Metered / low-bandwidth link	Any	4
Any	Any	Fewer hosts than threads	Set to the host count

Scanning is network-I/O-bound: each thread mostly waits on SSH/WinRM round-trips, not scanner CPU. Windows targets cost more per host than Linux (a first-run PowerShell warm-up tax). If raising the thread count makes "could not check" or timeout results climb, you have passed what the network or targets will tolerate — lower it.

Per-host risk profile

Each target carries a risk profile that filters which controls apply: laptop, workstation, server, dc. Auto-inferred from scan results and persisted at `DATA_DIR/target_profiles.json`; override via `PUT /api/target-profile/<target>`. DC profile skips IA.L2-3.5.6 and CM.L2-3.4.6 to avoid mid-remediation self-lockout.

Scan history & deltas

- `GET /api/scans` — paginated list of all scans
- `GET /api/scan/<id>` — full scan record
- `GET /api/scan/<id>/delta/<prev_id>` — structural diff (controls that changed MET/NOT MET; new findings; resolved findings)
- `GET /api/scan/history` — compliance score over time
- `GET /api/scan/<id>/csv | /pdf | /json` — export formats

Scan abort

`POST /api/scan/<id>/abort` signals an in-progress scan to stop after the current control completes per host — a graceful abort; results that did complete are persisted.

Persistent out-of-scope target list

Some IPs in your address space aren't customer endpoints and shouldn't appear in every discovery: build hosts, jump boxes, the operator's own workstation, network printers. Mark them out-of-scope once and ForteFide filters them out of every future Recon — Network, AD, and Jump Host modes all honor the list. Persistence is at `/var/lib/fortefide/excluded_targets.json` — separate from license.key so a dpkg-reinstall keeps exclusions even when other state is wiped.

8 · Remediation: Per-Target, Batch, Fleet

Remediation is the licensed operation that mutates a target to bring a NOT MET control into compliance. Every successful remediation writes a journal entry and a rollback record so the change can be reversed.

Three remediation modes

MODE	ENDPOINT	SCOPE	USE CASE
Per-target / per-control	`POST /api/remediate`	One control on one host	Surgical fix; investigating a single finding.
Batch (per-target multi-control)	`POST /api/remediate-batch`	All requested controls on one host	Bring one host to full compliance; sequential, ordered, journal-tracked.
Fleet (per-control across hosts)	`POST /api/remediate-fleet`	One control across N hosts	Roll out one fix to a fleet; parallel, seat-limit checked up front.
Async background job	`POST /api/remediate-async`	Same as batch, queued	Long-running multi-control jobs the operator wants to monitor without holding the HTTP request.

Remediation throttle & per-host progress

A Fix Threads slider (range 4-16, default 8) sets how many hosts to remediate at once. Tune it with the same scanner-hardware and network guidance as the scan throttle, but one step more conservative — each remediation worker holds an open privileged session and writes rollback records, so it is heavier per host than a read-only scan. A good default is to set Fix Threads at or below Scan Threads. The remediation API clamp is 1-20.

Remediation guarantees

- License-gated — every remediation endpoint verifies the license before queueing work; free tier returns 403 with an upgrade hint.
- Prepared-target gated — batch and fleet operations require the target be prepared (svc account in `_svc_accounts`).
- Seat-limit enforced — every remediate call verifies `len(_svc_accounts)` against the license cap up front.
- Smart ordering — controls within a batch are reordered to avoid self-lockout (e.g., do not change SSH config before deploying the new key).

- Manual-only filter — controls that cannot be auto-remediated are filtered before queuing and surfaced as `manual_required[]`.
- DC manual filter — on Windows targets profiled as DC, IA.L2-3.5.6 and CM.L2-3.4.6 are filtered (they change domain policy and risk locking out fortefide-svc mid-batch).
- Workers configurable — `/api/settings/remediation-workers` controls fleet parallel worker count. Default 4.

Per-control remediation outcomes

OUTCOME	WHAT HAPPENED	JOURNAL	ROLLBACK ELIGIBLE
success	Control re-evaluated MET; reversion data captured.	OPEN entry written.	Yes.
already_met	Pre-check found the control already MET; no change made.	No journal entry.	N/A.
manual_only	Control is in the catalogue but cannot be auto-remediated.	No journal entry.	N/A.
not_in_catalogue	Control has no remediation logic on this OS.	No journal entry.	N/A.
failed	Remediation ran but post-check did not show MET.	FAILED entry written for forensics.	Partial.
error	Remediation could not run (auth fail, command timeout, etc).	ERROR entry written.	No — nothing was changed.

Lockout-warning controls

Four Linux controls have a non-trivial chance of locking the operator out of the target if applied without preparation: **SC.L2-3.13.8** (FIPS-only cipher set), **AC.L2-3.1.8** (lock account on failed logins via faillock), **AC.L2-3.1.11** (SSH idle session timeout), **AC.L2-3.1.10** (session auto-lock timeout). The Remediate panel marks each with a lockout-warning badge and requires explicit acknowledgment via an ACK modal before they will queue:

```
IF THIS CHANGE IS MADE, YOU WILL LOSE ROLLBACK OPTION
(in the worst case where the new config also locks ForteFide out of the host).
```

- ACK is per-control, per-target — acknowledging once doesn't blanket-approve the same control on the next host.
- Without ACK, the remediate request returns 400 with `ack_required: true` and the list of unacknowledged control IDs.

- The four controls were singled out after live testing showed each could break SSH access via a different mechanism (cipher negotiation, faillock, ClientAliveInterval, MaxSessions). They are included in the Linux remediation catalogue's automated coverage of **67 of 110** controls — the ACK gate is what makes shipping them safe rather than shipping them at all.

Remediation history & evidence invalidation

`DATA_DIR/journal/remediation.jsonl` is a durable, append-only JSONL of every successful remediation. On startup the journal is replayed to rebuild `_remediation_history` — rollback survives a service restart. When a rollback fires, evidence collected from a pre-rollback scan is invalidated: `GET /api/evidence-status` returns `valid: false` with a reason. Rescan and regenerate evidence after rollback.

8a · Paired State-Change Records

ForteFide writes cryptographically-signed install/remove records for every state change it makes on a customer host. The pairing delivers what the leave-no-trace doctrine was always reaching for: not just operator-trust that ForteFide cleaned up after itself, but a verifiable cryptographic record an assessor can check offline.

What gets a paired record

- `svc_user` creation — `{username, os_type, auth_method}`
- `sudoers` edit (NOPASSWD entry)
- `ssh_key` deploy (scan_key.pub to authorized_keys)
- `ssh_cert` deploy (per-target 30-day cert)
- `ca_pubkey` deploy (`/etc/ssh/fortefide_ca.pub`)
- `sshd_config` edit (`TrustedUserCAKeys` directive)
- `watchdog` deploy (systemd unit)

Signing key derivation

HKDF-SHA256 from (license_half + binary_half) with a distinct salt/info from the SSH CA derivation, so the state-change key cannot collide with the CA seed even though the input keying material is the same. License rotation invalidates prior signatures by design — a new license is a new engagement.

Record format

```
{
  "version": 1,
  "ts": "2026-05-09T12:00:00+00:00",
  "engagement_id": "scan-42",
  "target": "10.0.1.50",
  "kind": "install" | "remove",
  "action_type": "svc_user" | "sudoers" | "ssh_key" | ...,
  "details": {...},
  "actor": "fortefide-26.07.27.1120",
  "sig": "<HMAC-SHA256 of canonical record bytes>"
}
```

Pair verification

Pair key: `(target, action_type, canonical(details))`. An install with no matching remove is a leave-no-trace audit gap, surfaced by `GET /api/state-changes/<engagement_id>` in the `unpaired_installs` array. Storage is append-only JSONL at `DATA_DIR/journal/state_changes.jsonl` with fsync after every write; records are never deleted in normal operation.

Third-party verification flow

- C3PAO assessor receives the signed evidence package + customer's license file
- Assessor extracts `state_changes.jsonl` from the evidence package
- Assessor re-derives the signing key from license_half (in license.key) + binary_half (from the FF binary)
- Assessor HMAC-SHA256s each canonical record and compares against the stored sig
- All sigs match + every install paired with a remove = leave-no-trace cryptographically proven — with no DenseDefense server access required

9 · Journal & Rollback

ForteFide's leave-no-trace promise is enforced by the journal. Every state change is recorded; rollback reads from the journal and never assumes anything beyond what was recorded. If ForteFide did not write it down, ForteFide will not undo it.

Journal entry shape

```
{
  "ts": "2026-05-04T14:23:11Z",
  "event": "remediate",   // or 'reverse', 'failed'
  "target": "192.0.2.20",
  "control": "AC.L1-3.1.1",
  "os_type": "linux",
  "reversion": { ...per-control reversion data... },
  "status": "open",       // 'open' | 'reversed' | 'failed'
  "scan_id": 17,
  "job_id": 4
}
```

Three rollback scopes

SCOPE	ENDPOINT	WHAT IT ROLLS BACK
Single	`POST` `/api/rollback`	One specific remediation entry (target + control).
Batch (per-target)	`POST` `/api/rollback-batch`	All open remediations against one target.
All (fleet-wide)	`POST` `/api/rollback-all`	Every open remediation across every prepared target, in reverse order per host.

- Per-control reversion data was captured at remediate time; rollback replays it, never re-derives it.
- If the target's current state already matches the pre-remediation state, the journal entry is marked reversed without re-applying.
- If a target is unreachable during rollback, the journal entry stays open for a later session to resume.

Live progress + honest counters

The activity log surfaces rollback progress per control, not just an end-of-run summary (e.g. "rollback 5/22 AU.L2-3.3.5 on 192.0.2.84..."). The summary line also distinguishes no-op from real failure:

RESULT MIX	LOG LINE	WHAT IT MEANS
All clean rollbacks	rollback-all on .X: 22 OK, 0 FAIL	Every control had remediation to undo; every undo succeeded.
All no-op	rollback-all on .X: 23 no-op (state already at baseline, nothing to revert)	Remediation never actually mutated state on this target — not a failure.
Mixed	rollback-all on .X: 18 OK, 3 no-op, 1 FAIL	Inspect the activity log entries above this line for the failing control's stderr.

Auto-rescan after rollback (unsigned verification)

When rollback-all finishes with at least one successful revert, ForteFide automatically fires a fresh authenticated scan against just those targets. This scan is marked **UNSIGNED** in the evidence package (`context=post_rollback_rescan, signed=False`) — it proves to the operator that rollback restored the baseline, but it is NOT C3PAO-grade evidence.

High-impact — Signed evidence is the artifact you ship to your C3PAO. Unsigned verification is for the operator's confidence that rollback worked — do not confuse the two; an assessor will reject any package whose manifest reports `signed=False`.

Journal hygiene

- `GET /api/journal/status` — open vs reversed vs failed counts, last-write timestamp
- `GET /api/reversion-bundles` — on-target reversion bundles ForteFide knows about
- `GET /api/reversion-bundles/<host>/<ts>/download` — one bundle for forensic review
- `GET /api/remediation-history` — rebuilt-from-journal history

Dashboard ROLLBACK button

A topbar ROLLBACK button carries a count badge of open journal entries — zero entries shows no badge; any open entries show a count, and clicking opens the rollback panel (single / batch / all). Stale rollback data from a prior session is surfaced via a separate banner with an explicit "review then act" prompt — never auto-rolled-back.

Note — Evidence packages signed before a rollback no longer reflect the machine's current state. `/api/evidence-status` returns `valid: false` after any rollback. Always rescan + regenerate evidence after rolling back.

Pre-flight reachability check

Before any rollback API call attempts reverse-commands over SSH/WinRM, `/api/rollback`, `/api/rollback-batch`, and `/api/rollback-all` probe the target's management ports with a 2-second timeout. If both are unreachable, the endpoint returns HTTP 503 pointing the operator at `/var/lib/fortefide/emergency-revert.sh` — not 30 silent SSH failures.

```
{
  "error": "Target X is unreachable on standard management ports...",
  "hint":  "Recovery option: log into the target's console as root
            and run `sudo /var/lib/fortefide/emergency-revert.sh`...",
  "code":  "target_unreachable",
  "probed_ports": [22],
  "success": false,
  "succeeded": 0, "failed": 0
}
```

9a · Uninstall Doctrine — License = Key = Proof

Uninstall is the deliberate end-of-engagement gesture. ForteFide treats it as a one-way exit governed by the leave-no-trace doctrine: scanner-side state goes away completely, and customer responsibility for evidence retention takes over.

Why the license file IS the trust anchor

Every signed artifact ForteFide produces — evidence package signatures, state-change records, SSH CA derivation — depends on HKDF-derived keys computed from license_half + binary_half. The derived keys are never persisted; they are re-computed when needed, and only when license.key is available. A customer who uninstalls without backing up license.key cannot re-derive the verification keys needed to validate kept-aside evidence packages — the packages still exist as files, but the cryptographic chain breaks.

Three operations, three behaviors

OPERATION	WHAT PERSISTS	WHAT CHANGES
Upgrade	Everything: license, journal, scan history, prepared svc accounts, evidence packages	Service stops, new binary installed, service starts; engagement continues
Uninstall	Nothing on the scanner (customer's external backups, if taken, preserve license + evidence)	PRERM gate prompts for explicit acknowledgment; POSTRM wipes `/var/lib/fortefide` and `/var/log/fortefide`
Reinstall (year later)	Nothing inherited — fresh license, fresh HKDF cycle, fresh state	Targets must be re-prepared from scratch

PRERM gate

When the .deb prerm script detects open journal entries, uninstall is refused by default. The operator must set `FORTEFIDE_UNINSTALL_ACK` to one of three explicit values:

- **rollback** — drain pending journal against affected targets; service accounts must still be reachable; long-running.
- **abandon** — `POST /api/journal/abandon` writes a signed abandonment record and sets `_evidence_invalidated`; targets keep changes; the audit trail records the deliberate choice.
- **cancel** — don't run the command at all; package stays installed.

No default — silent uninstall while pending state exists is the leave-no-trace failure mode. Default-deny + explicit-override is the correct policy for high-blast-radius cases.

After uninstall completes

Re-acquiring evidence requires the full process on a reinstall: license import → prepare endpoints → scan → remediate → generate evidence. There is no "refresh prior evidence" shortcut. Prior signed packages (if backed up) remain valid at their issue timestamp but require the backed-up license.key to re-verify.

9b · DDWitness — Cloud Stored Certified Evidence (Optional)

DDWitness is the OPTIONAL cloud destination for stored certified evidence packages. When configured, ForteFide encrypts a signed package locally and POSTs the ciphertext to a customer-scoped vault. C3PAOs are then granted engagement-scoped read access for evidence handoff.

Doctrine: license = key = proof (unchanged)

DDWitness stores AES-256-GCM ciphertext only. The encryption key is HKDF-derived from the customer's license file + the FF binary, on the scanner machine, and never leaves the scanner. DenseDefense / Cloudflare / the vault operator CANNOT decrypt. C3PAOs decrypt locally with the customer-supplied license.key + matching FF binary — same trust anchor as state-change signing, with a different domain separator so collision with other derived keys is impossible.

DDWitness is optional. The product still works 100% standalone without it. Airgap customers do not configure a vault; they produce the same signed package locally and hand it to the assessor via NIST 800-86 sneakernet. Customers who never enable DDWitness incur zero outbound network calls from the scanner.

Pricing

LICENSE TIER	DDWITNESS ADD-ON	SUGGESTED LIMITS
Free / Scanner-only	Not available	Sneakernet evidence handoff (free)
Licensed	$48/month per scanner	Up to 500 stored packages, 3-year retention, 5 assessor grants per package

Three operations

OPERATION	ENDPOINT	WHAT HAPPENS
Configure	`POST /api/vault/configure`	Stores the customer-issued vault upload token (mode 600). Never returns the token value.
Status	`GET /api/vault/status`	Reports configured state, vault URL, and the doctrine string.
Clear	`POST /api/vault/clear`	Removes the local token — does not revoke packages already in the vault.

`POST /api/scan/<scan_id>/vault-upload` triggers the full pipeline: build the signed evidence ZIP locally → derive the AES-256-GCM key via HKDF → encrypt the ZIP → POST ciphertext to the vault. When a customer uninstalls and acknowledges abandonment, the PRERM gate marks every package from that license as `proof_status='revoked'` — packages remain stored, but the DenseDefense warranty no longer attaches. DDWitness serves a customer + C3PAO web GUI where assessors paste a grant token (UUID) to download a scoped, expiring read of one package — decryption always requires the customer's license.key, delivered separately (no key escrow, ever).

10 · Evidence Collection & Signed Packages

Evidence is what the customer hands to a C3PAO assessor — the snapshot proof that, at a specific timestamp, on a specific machine fingerprint, with a specific license, the customer's fleet was in the recorded compliance state. ForteFide produces two evidence variants: draft (free-tier; unsigned) and signed (licensed; cryptographically anchored).

Evidence package contents

- `scan-summary.pdf` — compliance score, control breakdown, per-host results, per-family rollup
- `scan-results.json` / `.csv` — raw scan record + spreadsheet grid
- `control-evidence/` — per-control text/log capture as scanner observed it
- `host-inventory.json` — prepared targets, OS, auth tier realized, scan timestamps
- `remediation-history.json` — chronological remediation list from the journal
- `attestation/` — per-family signed attestation PDFs for the 31 organizational controls; controls in these families emit `ATTESTATION_REQUIRED` (not auto-assessed) so the denominator stays at 79 scanned controls
- `MANIFEST.sha256` / `.sig` — SHA-256 of every file + Ed25519 signature over the manifest (signed mode only)
- `license-fingerprint.json` — machine fingerprint + license tier + issued_at + expires_at (signed mode only)

Draft vs signed

PROPERTY	DRAFT (FREE)	SIGNED (LICENSED)
Endpoint	`GET /api/scan/<id>/evidence-package`	`POST /api/scan/<id>/signed-evidence-package`
Format	ZIP	ZIP with embedded MANIFEST.sha256.sig + license-fingerprint
Tamper detection	MANIFEST hash chain only	Ed25519 signature verifiable with DenseDefense public key
Identity binding	None	Bound to scanner machine fingerprint + license cycle
C3PAO ready	No — watermarked DRAFT	Yes — signature verifiable offline by assessor
Customer use	Internal review, pre-assessment dry runs	Submission to C3PAO; audit retention

Verifying a signed evidence package (assessor flow)

```
# 1. Verify the signature
openssl dgst -sha256 -verify densedefense_public.pem \
  -signature MANIFEST.sha256.sig MANIFEST.sha256
# Verified OK

# 2. Verify every file's hash
sha256sum -c MANIFEST.sha256

# 3. Confirm the machine fingerprint matches the customer's claim
cat license-fingerprint.json | jq .machine_fingerprint
```

Evidence after rollback

If a rollback runs after evidence is signed, `GET /api/evidence-status` returns `valid: false` with a reason string. The signed ZIP itself is still cryptographically valid — it accurately represents the machine state at signing time — but it no longer reflects the current state. Rescan and re-sign evidence after rollback before any C3PAO submission.

11 · The In-OS Watchdog

Remediation can fail in ways that leave the target in a partially-changed state — a config file rewritten before the validating syntax check fired, a service restart that hung. The in-OS watchdog arms during prepare and self-heals the target if remediation never recorded a completion heartbeat within the watchdog window.

Lifecycle

STAGE	WHAT HAPPENS
Arm (during prepare)	Phase 2 deploys a heartbeat script + cron entry on Linux targets. Windows targets rely on the hypervisor-snapshot safety net.
Pre-remediation snapshot	If a hypervisor adapter is configured (vSphere, Proxmox), ForteFide takes a snapshot before remediating; if not, the in-OS watchdog is the safety net.
Heartbeat refresh	Each successful remediation step refreshes the watchdog heartbeat so it stays quiescent.
Watchdog fires	If the heartbeat goes stale (default 15 min), the on-target watchdog reverts the host using the local reversion bundle.
Disarm	`POST /api/watchdog/<target>/disarm` tells the watchdog the operator is intentionally taking the host offline for maintenance.
Teardown	Removes the cron entry, the heartbeat script, and clears `_watchdog_state` for the target.

Inspecting watchdog state

- `GET /api/watchdog` — state for every prepared target
- `GET /api/watchdog/<target>` — one target
- `POST /api/watchdog/<target>/disarm` — temporarily disable self-heal during a planned outage
- `POST /api/watchdog/<target>/arm` — manually arm or re-arm the watchdog out-of-band, without rerunning Prepare

The asset scorecard's watchdog row exposes three explicit buttons — Arm, Disarm, Re-arm — alongside the last-armed timestamp and the most recent deploy result. A failed arm surfaces the `deploy_exception` string verbatim so the operator can act on the root cause.

Tip — The watchdog is a safety net for remediation runs that go sideways. It is NOT a continuous monitoring agent and not a replacement for hypervisor snapshots — where a hypervisor adapter is configured, ForteFide prefers the snapshot; the watchdog is the fallback.

11b · On-Machine Emergency Revert

The dashboard ROLLBACK button replays reverse-commands over the same SSH/WinRM channel that remediation may have just broken. When an SSH-hardening cohort wedges sshd on a Linux target, every reverse-command fails at the connection-refused step. The watchdog covers this case automatically; `emergency-revert.sh` covers the case where the operator wants to recover NOW, without waiting for the watchdog soak window, or where the watchdog itself can't fire (Windows target, manual disarm, snapshot missing).

What it is

A standalone shell script at `/var/lib/fortefide/emergency-revert.sh`, deployed to every prepared Linux target during Prepare Phase 2 alongside the watchdog. It restores the pre-engagement snapshot tarball, reload-first restarts sshd, verifies port 22 is listening within 10 seconds, disarms the watchdog so it doesn't fire a second revert, and prints a one-screen summary of what was restored.

How the customer uses it

Customer SSH access to the target is broken. They log into the target's console (vSphere, IPMI, iDRAC, BMC, physical keyboard) as root, and run:

```
sudo /var/lib/fortefide/emergency-revert.sh
```

No ForteFide running, no network, no hypervisor adapter, no customer-supplied credentials required. The script depends only on root + a working shell. After it completes, SSH access is restored and the customer can click ROLLBACK in the dashboard to flush the journal entries.

If the automatic watchdog timer detects a stale heartbeat and `emergency-revert.sh` is present on the target, the watchdog invokes it instead of doing a bare tar extract — the same verification + reload-first restart + port-22 smoke test as the manual recovery path.

> **Tip —** License = key = proof. Revert = local = guaranteed. The leave-no-trace doctrine has a recovery path that doesn't depend on the channel remediation could have broken, doesn't depend on hypervisor presence, doesn't depend on ForteFide running, and doesn't depend on the network being up.

Watchdog safety hardening

- Continuous heartbeat — the scanner pushes a heartbeat to every armed target every 5 minutes (not only at the end of a Remediate batch).

- Real disarm — `/api/watchdog/<target>/disarm` actually SSHes to the target and writes `soak_seconds=0`.
- `POST /api/watchdog/<target>/soak {"seconds": N}` for planned maintenance windows (bounded [60, 86400]).
- Boot-grace window: 300 seconds — planned reboots no longer trigger revert just because heartbeat mtime is stale.
- Snapshot HMAC — integrity-verified before extract; tampered snapshots refuse to restore and log the mismatch.
- Mid-batch lock — `watchdog.suspend_until=now+4h` written at Remediate batch start; cleared at successful completion.

Journal-driven per-control reverse

The watchdog has two undo paths. The broad path extracts a snapshot tarball of critical config paths over `/` — works for the typical "broke sshd_config" case but also overwrites legitimate post-Prepare sysadmin edits to those paths, and doesn't cover paths the snapshot didn't capture. The precise path runs the catalogue reverse-command for each remediated control: at every successful Apply, ForteFide deploys the reverse command to `/var/lib/fortefide/reverses/<control_id>.sh` (mode 0600) and appends a target-stamped JSON line to `journal.jsonl`. On stale heartbeat, the watchdog reads the journal, builds a LIFO list of unpaired applies, and runs each reverse script in order — only what ForteFide actually applied gets reversed.

Mix-mode safety net: controls whose catalogue rollback is `rollback_safe=False` (e.g. AC.L2-3.1.6's root password lock, irreversible without console access) skip the per-control path and fall through to the snapshot-tarball restore.

Firewall remediation: signed operator acknowledgment required

Any firewall-touching remediation requires an explicit signed operator acknowledgment before ForteFide will auto-Apply. Affected controls (same IDs on Linux + Windows):

- AC.L1-3.1.20 — Install + enable host firewall
- SC.L1-3.13.1 — Install + enable host firewall (paired with AC.L1-3.1.20)
- SC.L2-3.13.6 — Deny-all-by-default + permit-by-exception
- SC.L1-3.13.5 — Verify host firewall active

Firewall changes are the highest-blast-radius remediations ForteFide runs — a misapplied rule can lock the operator out (covered by the journal-driven watchdog reverse) and lock out every customer process that depends on the network (not covered by the watchdog). Operator workflow: hit `POST /api/firewall-ack` with `operator_name` + `justification` (>=10 chars) before running Remediate. ForteFide HMAC-signs the ack with the license-derived state-change key and persists it; the Remediate engine verifies the signature before executing each firewall control. Acknowledgments bundle into the

signed evidence ZIP so the C3PAO sees the explicit operator authorization for every firewall change ForteFide made.

12 · Customer Action Required

Customer Action Required (CAR) is the channel ForteFide uses to ask the customer for help when ForteFide genuinely cannot proceed without operator intervention. CAR is intentionally narrow: it fires only when a specific package required for a specific operation cannot be obtained from any available install layer.

When CAR fires

Layer 1 is default-bundled in the customer .deb. CAR is a true last-resort signal — it fires only when all three layers genuinely fail. For the supported distro+version combinations, the customer never sees a CAR prompt: Layer 1 covers them in the bundle, with Layer 3 (target's own apt/dnf repos) and Layer 2 (mounted install media) used opportunistically.

- Recon detected the target's distro/version
- ForteFide needs a package for the next operation (scan / remediate / revert)
- The package is not already installed on the target
- Layer 3 (target's own package manager / repos) failed — no network, no local mirror
- Layer 2 (customer's mounted install ISO) is not present or does not contain the package
- Layer 1 (ForteFide's vendored bundle) does not have a bundle for this distro+version, or the bundle is missing the specific package

Only when ALL of the above hold does CAR fire.

CAR endpoints

- `GET /api/customer-action` — list of all targets with pending CAR (empty in normal operation)
- `GET /api/customer-action/<target>` — one target's CAR state
- `POST /api/customer-action/<target>/retry` — customer signals they have mounted install media; ForteFide re-probes for the package

Operator response to CAR

CUSTOMER REALITY	RIGHT ANSWER
Network-connected with internet access	Disable airgap mode — Layer 3 will resolve; CAR will not fire.
Air-gapped, package is on the customer's install ISO	Mount the ISO at `/mnt/install-media` and POST the retry endpoint.
Air-gapped, install ISO not available, bundle is missing the package	Contact DenseDefense to publish a bundle update, or install via the customer's usual offline channel and retry.
Distro outside ForteFide's supported set (e.g. AIX, Solaris)	Out of scope — document as out-of-scope in the assessment boundary.

Tip — CAR is a tool for explicitly air-gapped environments. The default deployment model is recon-and-scan-what's-present — ForteFide assesses the customer's environment as it stands, and asks for help only when a specific operation actually needs an unavailable package.

13 · Airgap Installation: The 3-Layer Model

ForteFide is designed for air-gapped CMMC enclaves. The same binary works whether the scanner has internet, the targets have internet, or neither does. Package resolution falls through three layers in priority order.

LAYER	WHAT	WHERE	CUSTOMER EFFORT
3	Target's own package manager + repos	apt / dnf / zypper / pacman over network or local mirror	None — default behavior.
2	Customer-supplied install media	Mount ISO/DVD at `/mnt/install-media` (Linux) or analogous Windows mount	Last-resort only — mount when prompted by CAR, then retry.
1	Vendored bundles inside the ForteFide binary	`/opt/fortefide/packages/<family>/<version>/` — shipped by default in the customer .deb	None — already on disk after install.

Tip — The customer .deb ships with `packages/` pre-populated for all supported distro+version combinations by default. Tooling-only `--no-airgap` builds are still available for build hosts and CI runners but are NOT the customer-shipping default. The CAR prompt fires only when Layer 1 lacks a bundle for the target's distro AND Layer 3 is unreachable AND Layer 2 is empty.

What Layer 1 contains

Layer 1 ships two scoped categories of packages. It does NOT ship distro install images, kernels, or arbitrary user-space tools.

- **Category A — Auth cascade enablers:** openssh-server (per distro: .deb, .rpm, .pkg.tar.zst) plus a thin layer of mandatory deps. Without Category A, ForteFide cannot reach the target at all — it is the load-bearing prerequisite.
- **Category B — CMMC remediation kit:** the family of packages remediation engines might push (aide, auditd, apparmor, clamav, libpam-pwquality, rsyslog, ufw / firewalld, lynis, fail2ban, etc). These are NOT prerequisites for scanning — pushed only during an explicit remediate operation, with operator consent.

Bundle verification

Every Layer 1 bundle ships with `MANIFEST.sha256` (per-file SHA-256) and `MANIFEST.sha256.sig` (Ed25519 over the manifest). At runtime, `airgap_packages.verify_bundle()` validates both before the bundle is consulted. Tampered bundles, bad signatures, or extra unsigned files all cause ForteFide to refuse the bundle.

Recon-and-need-driven activation

Layer 1 is on disk after install but is NOT automatically used during scans. The fallback chain only walks down to Layer 1 when both gating conditions hold: recon confirmed the target's distro, and the target lacks a specific package ForteFide needs for the specific operation it is about to perform. In the normal case (network-connected fleet, target already has openssh-server), Layer 1 is never consulted.

> **Tip —** What ForteFide will NOT do: it does not configure the customer's environment before scanning, does not demand the customer install prerequisites, and does not refuse to scan a target whose OS is not in the bundle catalogue. Doctrine: recon, then scan what is present.

14 · Logs & Troubleshooting

Log locations

LOG	LINUX PATH	WINDOWS PATH
Audit log (append-only)	`/var/log/fortefide/fortefide.log`	`%ProgramData%\ForteFide\logs\fortefide.log`
Activity log (in-memory ring)	`GET /api/logs` (last 1000 entries)	`GET /api/logs`
Activity export	`GET /api/logs/export` (CSV)	`GET /api/logs/export`
Journal (durable)	`/var/lib/fortefide/journal/remediation.jsonl`	`%ProgramData%\ForteFide\journal\remediation.jsonl`
Scan database	`/var/lib/fortefide/scan.db` (encrypted SQLite)	`%ProgramData%\ForteFide\scan.db`
Reversion bundles (on target)	`/var/lib/fortefide/reversion-bundle.json`	`%ProgramData%\ForteFide\reversion-bundle.json`

The audit log defaults to INFO level. Run the systemd unit with `--debug` to elevate to DEBUG — noisier, but the right tool when diagnosing prepare or remediation failures.

Common failure modes

PREPARE ABORTS AT "CREATING SERVICE ACCOUNT"

Symptom: prep_id status reports failure with stderr containing "sudo: a password is required." Cause: the bootstrap user lacks NOPASSWD sudo and Phase 0's write was rejected (sudoers.d disabled, immutable bit, etc). Fix: pre-grant NOPASSWD via your normal channel and retry prepare.

TARGET WENT DARK ON PORT 22 AFTER SSH-HARDENING REMEDIATION

Symptom: post-remediation scan shows the target unreachable on port 22; manual `nc` returns "Connection refused." Cause: the SSH-crypto cohort appended Ciphers/MACs/KexAlgorithms to sshd_config and restarted sshd; if the appended config contained a cipher the host's sshd build doesn't support at runtime, sshd_config was syntax-valid but the daemon failed to start. The SSH-crypto-harden path uses reload-before-restart with a 2-second post-action smoke test that verifies port 22 is still listening — if the smoke test fails, the command automatically restores `sshd_config` from the baseline and bounces sshd again, so you will never get locked out. If the target stays unreachable anyway, console-recover via `emergency-revert.sh` (see §11b).

REMEDIATION REPORTS "PACKAGE_UNAVAILABLE" (KALI / AIRGAP)

Symptom: a control fails with stderr containing "package not installed and not available from any configured repository." Cause: the control needs rsyslog or auditd installed, but the target's package manager couldn't fetch it — common on Kali (no rsyslog/auditd in the base install) and air-gapped hosts where Layer 1 lacks the specific package. Fix: install the package on the target beforehand, or ensure Layer 1 contains the package for the target's distro.

TIER 1 CERT AUTH NEVER LANDS; EVERYTHING FALLS TO TIER 2

Symptom: `/api/svc-accounts` shows `auth_method=ssh_key` for every host. Cause: license issue (HKDF derivation failed) or scan_key generation failed (`POST /api/generate-scan-keys` to force regeneration). Falling back to Tier 2 is supported — the product still works.

"ENDPOINT NOT PREPARED" ON REMEDIATION

Symptom: `POST /api/remediate-batch` returns 400 with "Endpoint not prepared. Run /api/prepare-endpoint first." Cause: the target is not in `_svc_accounts` — prepare was never run, prepare failed, or (in an old build) auto-teardown wiped the account. Fix: run prepare on that host.

SCAN HANGS ON A HOST

ForteFide does not run port scans separately. If a scan hangs on a specific host, the issue is auth or network — `POST /api/scan/<id>/abort` to abort gracefully; completed results are persisted.

AD DISCOVERY FAILS WITH NTLM / MD4 / "UNSUPPORTED" ERROR

Symptom: `POST /api/recon-domain` mentions MD4, NTLM, or "unsupported hash" on a modern Linux scanner host. Cause: OpenSSL 3.0+ disables the legacy MD4 hash by default; LDAP libraries that default to NTLM bind require MD4. ForteFide auto-falls back to a SIMPLE LDAP bind when MD4/NTLM is unavailable — transparent and logged. If SIMPLE bind also fails, the credential is genuinely wrong.

The failure-report endpoint

`GET /api/failure-report` aggregates failure signals from the journal, state changes, watchdog heartbeat history, and the activity log into one classified report. The dashboard's topbar REPORT button surfaces it; the standalone tool `lab_tools/failure_report.py` runs the same classifier against on-disk artifacts without requiring the FF service to be running.

CATEGORY	WHAT IT MEANS
auth_failure	SSH / WinRM auth rejected.
network_unreachable	TCP connect to 22 / 5985 / 5986 timed out or was refused.
sudo_password_required	Bootstrap user lacks NOPASSWD; Phase 0 cannot proceed.
package_unavailable	Layer 3 + 2 + 1 all failed for a needed package; CAR fires.
watchdog_deploy_exception	Watchdog cron/script deploy raised during Prepare.
watchdog_fired	Heartbeat went stale; the on-target watchdog reverted the host.
remediation_step_failed	A remediation step returned non-zero.
scan_timeout	A scan job did not produce results within the per-host budget.
evidence_sign_failed	Signature derivation failed (license missing, expired, or HKDF input corrupted).
stale_rollback_data	Journal entries from a prior session present at fresh-install/restart — surfaced via banner, never auto-rolled-back.
unclassified	A failure that didn't match any of the patterns above.

15 · Known Issues & Roadmap

ISSUE	STATUS	WORKAROUND	TARGET FIX
Windows teardown can timeout intermittently	sshd is transiently unavailable on Windows targets immediately after the rescan phase, causing teardown to time out at ~20 sec. Linux teardown drains cleanly under the same conditions.	Retry the single host via `/api/teardown-endpoint` after a 30 sec pause; `/api/teardown-batch` isolates per-host failures.	Pre-flight port-22 smoke test on every Windows target before teardown.
Dashboard Fix-All job count is non-deterministic	Walks an in-memory findings collection that may be partially populated when the click fires.	For scripted workflows, drive remediation per-host via `/api/remediate-batch` for deterministic job counts.	Server-side deterministic remediation plan endpoint from a scan_id (in progress).
No single authoritative engagement-state endpoint	Consumers infer phase by polling per-job state.	Poll `/api/remediate-job/<id>` with `current_control == null` as the authoritative completion signal.	Candidate: `GET /api/engagement/<id>/status` returning a single phase + counts payload.
Arch Linux: limited catalogue coverage	Generic Linux catalogue can brick Arch boxes (PAM/sshd/audit/firewall conventions diverge from RHEL/Debian assumptions).	Validate Arch endpoints in a sandbox before remediating production; use scan-only mode if uncertain.	Arch catalogue branch in flight; will be advertised as supported only when end-to-end validated.
Scheduler endpoints marked Beta	The endpoints exist but the schedule loop is a placeholder.	Use cron / Task Scheduler with `curl` against the scan endpoint in the meantime.	Full scheduler implementation.
Hypervisor snapshot integration is single-adapter	vSphere supported; Proxmox in flight. Where no adapter is configured, the in-OS watchdog is the only safety net.	Run remediations in batches small enough that the watchdog window (default 15 min) is sufficient.	Continuous coverage as adapters are added.

16 · Security Architecture

ForteFide operates on customer-owned production machines under operator authorization. The security model below describes what trust ForteFide claims, where the trust comes from, and what the customer can verify independently.

Trust roots

TRUST ROOT	WHERE IT LIVES	USED FOR
DenseDefense Ed25519 publisher key	Hard-coded in the binary; published at densedefense.com/keys	Verifying license.key signatures; signed evidence manifests; Layer 1 bundle manifests.
License HKDF seed	Embedded in license.key (encrypted at rest under the machine fingerprint)	Deriving the per-scanner SSH CA keypair — each license produces a distinct CA.
Per-scanner SSH CA	Derived in memory; private key wiped after each prepare; public key persisted to targets	Issuing 30-day SSH certs to fortefide-svc; cert principal = fortefide-svc; cert validity = 30 days.
Per-target reversion bundle	On the target	The authoritative record of what ForteFide changed; drives teardown and rollback.

What ForteFide protects against

- **Tampering with evidence packages** — signed mode embeds an Ed25519 signature over the MANIFEST and binds the package to the scanner's machine fingerprint; any byte-level change invalidates the signature.
- **Cross-customer evidence forgery** — each license produces a distinct SSH CA, and license-fingerprint.json binds the package to a specific scanner machine fingerprint.
- **Stale credential reuse on retired hosts** — per-target certs expire after 30 days.
- **License sharing across machines** — copying license.key to another scanner does not activate it; `/api/import-license` rejects mismatched fingerprints.
- **Tampered Layer 1 bundles** — MANIFEST.sha256 + Ed25519 signature; tampered bundles are refused.
- **Privilege escalation by service account** — fortefide-svc access is removed by Teardown; operator admin credentials are not stored beyond Phase 0/1/2.

What ForteFide does NOT protect against

- A compromised scanner host — operator hygiene is the customer's responsibility.
- A compromised target during the engagement — ForteFide scans state; it cannot detect a sufficiently stealthy rootkit that lies to userland tools.
- Modification of the customer's machine before Recon — ForteFide reports against the state it observes.
- Misuse of the Ed25519 publisher key — key rotation policy is documented in the C3PAO Guide.

Encryption at rest

- `scan.db` — AES-256-GCM, key derived from the machine fingerprint.
- `license.key` — the public-payload portion is plaintext; the HKDF seed is encrypted under the machine fingerprint.
- Per-target cert keys — cleartext on disk under `/var/lib/fortefide/certs/<ip>/key` (mode 0600) — replaceable artifacts, so no additional encryption layer.
- Journal — cleartext JSONL; contents are non-sensitive and need to be human-readable for forensic review.

Network footprint — one connection per target

Both scanning and remediation against a Linux target run through ONE already-authenticated SSH session per (engagement, target). The session covers preflight, every control execution, every post-control liveness probe, every auto-rollback fire, and the post-remediation verification scan — then closes. A 79-control scan plus a remediation pass on the same target produces exactly one TCP handshake on port 22. This shape matters: it avoids tripping sshd's `MaxStartups` rate limit, avoids enterprise firewall connection-rate rules, and keeps the existing session alive through an sshd restart that a hardening control might trigger. (Windows targets retain per-call WinRM sessions; the same refactor for Windows is on the roadmap.)

Auth-tier cascade + admin-cred fallback

Every scan tries the available authentication tiers in order, falling through on failure. After all three service-account tiers fail, ForteFide escalates one more step: it retries the scan using the engagement-level admin credential supplied at Prepare time. This exists because hardening controls applied during Remediation occasionally invalidate the service account's deployed cert or NTLM acceptance (cert thumbprint mapping changes, GPO refresh, NTLMv2-only enforcement). Without the fallback, a post-state verification scan that couldn't auth as the service account would render the host as N/A, hiding every control that was just successfully remediated.

When the fallback fires: the activity log logs "ESCALATING to admin-cred fallback"; the asset scorecard renders with an orange border + "Verified-by-Admin" label (a context tag, not a quality discount); evidence package entries carry `verified_via_admin_fallback: true` with a `fallback_reason`. The service-

account abstraction is preserved for Prep, Remediate, and Teardown — the fallback is scope-bounded to scan/verification only.

Clear History — between-engagement state wipe

The CLEAR HISTORY button wipes the scanner's in-memory engagement state — scans, discovered/imported inventory, prepared service-account records, watchdog and CAR state, and the activity log. The license file is preserved. Two-step confirmation warns that clearing without Teardown orphans the service accounts on those hosts; the right sequence is Teardown first, then Clear History. Backed by `POST /api/session-clear` (`confirm="CLEAR_ALL"` , optional `force_orphan=true`).

17 · Maintenance, Lifecycle & Backup

ForteFide is intended to live across multiple compliance engagements on the same scanner. This section covers between-engagement maintenance, artifact rotation, and backup/restore for license continuity.

Engagement vs license cycle vs install lifetime

CONCEPT	BOUNDED BY	WHAT ROTATES
Engagement	Operator-defined: typically one scan + remediate + evidence cycle; days to weeks	fortefide-svc accounts (created at start, removed at Teardown); per-target reversion bundles.
License cycle	License expiry (typically 1 year, tier-dependent)	Seat tally; signed evidence packages bind to the license fingerprint at signing time.
Install lifetime	From `dpkg -i` to `dpkg --purge`; survives upgrades	scan_key (Tier 2 SSH key); SSH CA derivation seed (license-bound); historical scan database.

Routine maintenance tasks

- **Weekly:** review `GET /api/journal/status`; reverse any "failed" entries no longer needed; confirm no stale `_svc_accounts` from incomplete teardowns.
- **Monthly:** review `GET /api/license-status` days_remaining; plan renewal at >=30 days remaining; review `GET /api/scan/history` for compliance trend.
- **Quarterly:** rotate scan_key (`POST /api/generate-scan-keys`) and re-prepare each target on next engagement; audit `/var/lib/fortefide` for stale entries.
- **Per release:** read the changelog; upgrade with `dpkg -i` (preserves `DATA_DIR`); test prepare against a non-production target before fleet-wide use.

License renewal

Licenses are dated. `/api/license-status` reports days_remaining; the dashboard surfaces a banner under 30 days. Renewal is the same flow as initial activation. Evidence signed under the old license remains verifiable indefinitely.

> **Tip —** The signed evidence package binds to the license fingerprint AT SIGNING TIME, not to the currently-active license. The Ed25519 signature was made with the DenseDefense publisher key, which is stable across renewals — an assessor gets the same answer in year 1 and year 6.

Scanner migration (move to new hardware)

- On the new scanner: install ForteFide; record the new request code.
- Contact DenseDefense support with the old license info + new request code — the old license is invalidated server-side; a new one is issued bound to the new fingerprint.
- Drag-drop the new license.key on the new scanner.
- All previously-prepared targets need to be re-prepared from the new scanner — the new license derives a new SSH CA; old certs do not carry over.

Backup

CATEGORY	FILES	RECOMMENDED CADENCE	NOTES
Critical	`DATA_DIR/license.key`	Immediately after import; before any reinstall/uninstall	Loss = need to contact DenseDefense for re-issue.
Important	journal, scan.db, target_profiles.json, _svc_accounts.json	Weekly during active engagements; daily during a remediation campaign	Loss = lose remediation history and scan history.
Operational	certs/, keys/scan_key, signed evidence packages already produced	Per release of the customer journey	Loss of certs/scan_key = re-prepare each target (regenerable).

Restore

```
# Stop the service
sudo systemctl stop fortefide

# Restore from your backup
sudo tar -xzf fortefide-data-backup.tar.gz -C /

# Verify ownership (defaults to root:root)
sudo chown -R root:root /var/lib/fortefide

# Restart the service
sudo systemctl start fortefide
```

Uninstallation (clean)

The uninstall path is GATED: the package's prerm script requires the `FORTEFIDE_UNINSTALL_ACK` environment variable set explicitly to either "rollback" or "abandon" — the License = Key = Proof doctrine enforced in code.

```
# 1. Back up the license (CRITICAL -- this is the trust anchor)
sudo cp /var/lib/fortefide/license.key ~/license.key.bak
```

```
# 2. Optional: also back up signed evidence packages
sudo tar -czf ~/fortefide-data-backup.tar.gz \
  /var/lib/fortefide /var/log/fortefide

# 3. Stop AND disable the service (disable --now, not just stop)
sudo systemctl disable --now fortefide.service

# 4. Purge with the ACK env var. Pick ONE:
sudo FORTEFIDE_UNINSTALL_ACK=abandon dpkg --purge fortefide-scanner
# OR (only if all targets still reachable for journal drain):
sudo FORTEFIDE_UNINSTALL_ACK=rollback dpkg --purge fortefide-scanner

# 5. Verify
ls -la /var/lib/fortefide   # should not exist
ls -la /opt/fortefide       # should not exist
dpkg -l | grep fortefide    # should show no installed package
```

If you forget the `FORTEFIDE_UNINSTALL_ACK` env var, prerm exits with an acknowledgment-required message and the package lands in "pi" (purge-incomplete) state — recover by re-running the purge command with the ACK env var set.

Customer-side cleanup obligations

Removing ForteFide from the scanner does NOT clean up prepared targets. Always run Teardown on every prepared host before uninstalling the scanner.

- Run `/api/teardown-endpoint` on every host in `/api/svc-accounts`
- Confirm `/api/svc-accounts` is empty before `dpkg --purge`
- On any host you cannot reach for teardown, manually remove the artifacts listed in Appendix C

18 · Operational Playbooks

Six end-to-end playbooks for the most common ForteFide engagements. Use these as the starting point for SOPs in your own environment.

Playbook 1: First-time scan of a new fleet (no remediation)

Goal: get a compliance baseline across a fleet you've just inherited. No license required.

1. Open the dashboard, click Recon. Enter the CIDR covering your fleet. Click Discover.
2. Wait for discovery; review the reachable hosts list.
3. Skip Prepare. Click Scan; choose "Password scan"; enter admin credentials; choose Full profile.
4. Watch the scan dashboard — 3-8 min/host.
5. Review results; the compliance percentage and per-family breakdown are your baseline.
6. Generate Evidence Package; download the DRAFT (unsigned) ZIP for internal review.
7. Done — no teardown needed since nothing was created on the targets.

Playbook 2: Full readiness engagement (Starter / Pro)

Goal: take a fleet from baseline to assessment-ready, sign evidence, then leave the fleet exactly as you found it.

1. Recon (CIDR or host list).
2. Prepare — select all in-scope hosts, auth method = Certificate (default), enter admin credentials; expect 1-3 min/host.
3. Scan — select prepared hosts, profile = Full; auth happens at Tier 1 automatically.
4. Review baseline — note the pre-remediation compliance percentage.
5. Remediate — select controls (or "Remediate all auto-fixable"); Per-target, Batch, or Fleet.
6. Re-scan (Quick profile) — verify controls moved MET; note the post-remediation score.
7. Generate Evidence Package; choose Signed; download the ZIP + .sig.
8. (Optional) hand the signed ZIP to the C3PAO.
9. Teardown — click Teardown All Prepared; confirm `/api/svc-accounts` is empty.

Playbook 3: Single-host investigation

Goal: a specific host failed a finding; investigate and fix without disturbing the rest of the fleet.

1. Recon just the host (or use an existing discovery record).
2. Prepare just the host.

3. Scan just the host (Full profile).
4. Click into the failing control; review evidence (stdout, stderr, exit code, expected vs actual).
5. Remediate that one control via `POST /api/remediate`.
6. Quick re-scan that one control via `/api/matrix/<control_id>`.
7. Teardown the single host.

Playbook 4: Roll out one fix across the fleet

Goal: a newly-identified weakness needs to be fixed across many hosts in one wave.

1. Confirm prepared state on all targets: `GET /api/svc-accounts`.
2. `POST /api/remediate-fleet` with the target list and the control_id — ForteFide pre-checks the seat limit.
3. Poll `GET /api/remediate-fleet/<job_id>` — default 4 parallel workers.
4. Review failures; re-target failed hosts individually after fixing the root cause.
5. Quick re-scan to verify on each remediated host.
6. Generate Evidence Package only after all remediations are verified.

Playbook 5: Roll back a remediation that broke something

1. Identify the journal entry: `GET /api/journal/status` to confirm it's still "open".
2. Decide scope: single, batch, or all.
3. `POST /api/rollback` / `/api/rollback-batch` / `/api/rollback-all`.
4. Watch the log — each per-control reversion logs as reversed.
5. Check `/api/evidence-status` — if you'd already signed evidence, plan to rescan + re-sign before any C3PAO submission.
6. Decide whether to re-attempt the remediation or accept the finding for now.

Playbook 6: Air-gapped deployment

1. On a connected build station, download the customer .deb (Layer 1 bundles included by default).
2. Transfer the .deb via approved removable media.
3. Install (`sudo dpkg -i`); pre-accept EULA.
4. Generate a request code on the (offline) scanner; transfer it out via an approved channel.
5. Purchase a license on densedefense.com from a connected workstation; download license.key.
6. Transfer license.key into the enclave; drag-drop it into the License panel.
7. Enable airgap mode in the dashboard's operational settings.

8. Run Recon, Prepare, Scan. If a target needs a package not on Layer 1, mount the install ISO and POST the CAR retry endpoint.

19 · Per-Distro Prepare Notes

Prepare's Phase 0 NOPASSWD step is the most distro-sensitive part of ForteFide's flow. The notes below cover the validated set; new distros tend to follow one of these patterns.

Debian / Ubuntu

- Cloud-init users ship with NOPASSWD via `/etc/sudoers.d/90-cloud-init-users` — Phase 0 is a no-op.
- Non-cloud-init installs: Phase 0 writes `/etc/sudoers.d/<user>-nopasswd`.
- openssh-server is on by default in cloud-init images — Phase 0+ is a probe-only no-op.
- Validated: Debian 11, Debian 12, Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04.

RHEL / Rocky / Alma

- Cloud-init users typically have NOPASSWD via cloud-init-supplied sudoers; Phase 0 verifies and writes the override if missing.
- SELinux defaults to Enforcing on most cloud images — ForteFide does not change SELinux mode; control checks report current state.
- Validated: RHEL 7 (legacy support), RHEL 8, RHEL 9, Rocky 8, Rocky 9, Alma 8, Alma 9.

SUSE / SLES

- SLES has `Defaults targetpw` (sudo expects root's password) — Phase 0 writes `Defaults:<user> !targetpw` + NOPASSWD.
- openssh-server present; service may be named sshd (systemd) — Phase 0+ ensures enabled and started.
- Validated: SLES 12 SP5 (legacy), SLES 15 SP4/SP5, SLES 16 (rolling). SLES 12 may need Layer 2 fallback for some remediation packages (Lynis is in Package Hub, not a standard mirror).

Arch (rolling)

- Cloud-init for Arch is not always wired up cleanly; bootstrap user NOPASSWD must be present in `/etc/sudoers.d/` before Prepare can run.
- Lab convention: drop `/etc/sudoers.d/10-scanuser` with `'scanuser ALL=(ALL) NOPASSWD: ALL'`.
- Arch's PAM/sshd/audit/firewall conventions diverge from Debian/RHEL assumptions; the generic Linux remediation catalogue can brick Arch boxes — validate in a sandbox before remediating production.
- Validated: limited test coverage (full Prepare path exercised; scan-only otherwise).

Windows Server / Windows 10 / Windows 11

- Windows targets need both OpenSSH Server and WinRM available. Prepare uses WinRM for Phase 1 (account creation) and SSH for the steady-state.
- Server 2012 R2: no `Get-LocalUser` cmdlet — ForteFide falls back to `net user` / `net localgroup`. Cert auth holdout on 2012 R2 is documented in §15.
- Server 2016+ / 10 / 11: full path supported.
- Domain Controllers: ForteFide profiles them as "dc" and auto-skips IA.L2-3.5.6 + CM.L2-3.4.6 to avoid mid-batch self-lockout via domain policy changes.

20 · Control Family Deep Dive

Per-family overview of what ForteFide actually does for each of the 14 NIST 800-171 Rev 2 control families. This is ForteFide's behavior, not the framework definition — consult NIST SP 800-171 Rev 2 for the requirement text.

AC — Access Control (22 controls)

Checks: account configuration, session controls, least-privilege, remote access, and information flow. Auto-checkable: 22. Risk on remediation: medium — changing SSH config or firewall rules can lock out access; smart ordering deploys new keys before changing SSH config.

AT — Awareness & Training (3 controls)

Entirely policy / process. Attestation templates let the operator mark these MET with supporting documentation.

AU — Audit & Accountability (9 controls)

Checks: auditd/journald enabled and configured, retention, privileged-operation audit trail, time synchronization, log review processes. Auto-remediation: enable auditd, deploy stock audit rules, configure logrotate, enable Windows audit policies via auditpol. Risk: low.

CA — Assessment & Authorization (4 controls)

Assessment process controls. Auto-checkable: 2. Manual: 2 (organizational process).

CM — Configuration Management (9 controls)

Checks: baseline config files, package install enforcement, least functionality. Auto-checkable: 9; CM.L2-3.4.6 is DC-aware (applies on a DC, skipped on a domain member). Risk: medium-high — disabling services on a DC can break domain operations, handled by the DC profile filter.

IA — Identification & Authentication (11 controls)

Checks: password policy, MFA presence, unique user IDs, replay-resistant auth, identifier management. Auto-checkable: 11; IA.L2-3.5.6 is DC-aware. Risk: high on DC — password policy changes propagate fleet-wide via GPO; the DC profile filter exists for exactly this risk.

IR — Incident Response (3 controls)

ForteFide supports these via audit trail and evidence retention; the IR plan and exercises are organizational. Auto-checkable: 2. Manual: 1.

MA — Maintenance (6 controls)

Checks: maintenance personnel access, tool usage, off-site maintenance, timely patching. Auto-checkable: 5. Manual: 1.

MP — Media Protection (9 controls)

Checks: removable media handling, encryption-at-rest (LUKS/BitLocker presence), media sanitization readiness. Auto-checkable: 8. Manual: 1 (process/physical). USB blocking is enforceable via udev (Linux) or Group Policy (Windows); ForteFide checks current state but leaves enforcement to your platform tooling.

PE — Physical & Environmental (6 controls)

Entirely physical. Manual attestation only.

PS — Personnel Security (2 controls)

Entirely process. Manual attestation only.

RA — Risk Assessment (3 controls)

Checks: vulnerability-scan presence/cadence and patch/update posture as technical evidence for the risk-assessment process. Auto-checkable: 3.

SC — System & Communications Protection (16 controls)

Checks: cryptographic configuration, boundary protection, session protection, separation of duties. Auto-checkable: 15. Manual: 1. Risk: medium — changing SSH cipher suite can break older clients; ForteFide's catalogue uses modern-but-compatible cipher choices and falls back if validation fails.

SI — System & Information Integrity (7 controls)

Checks: AV/EDR presence, AIDE/Tripwire presence (Linux), intrusion detection, security alerts process, vulnerability scanning. Auto-checkable: 7. AV signature freshness is checked but ForteFide does not ship clamav-data; the customer's internal mirror or external sig-update process handles freshness.

21 · Sample Control Catalogue

A curated sample of how ForteFide checks and remediates specific NIST 800-171 controls, drawn from the full 110-control catalogue. This sample picks one representative control per family (where auto-checkable) plus the highest-risk controls operators should understand before remediating.

CONTROL	WHAT FORTEFIDE CHECKS	AUTO-REMEDIATES?
AC.L1-3.1.1	Authorized users only — /etc/passwd review on Linux; Local + Domain Users on Windows.	Partial — can disable known orphaned accounts; manual review for ambiguous cases.
AC.L2-3.1.2	Authorized transactions / functions — group membership, sudoers, Administrators/Domain Admins audit.	Manual — too policy-dependent.
AC.L2-3.1.9	Notification at logon — /etc/issue.net banner; Windows logon banner registry keys.	Yes — deploys stock CMMC banner with rollback.
AC.L2-3.1.7	Privileged functions audit — sudo log review; auditd rules; Windows Privileged Audit Policy.	Yes — deploys auditd rule set; configures Windows auditpol.
AC.L2-3.1.19	Inactive session termination — ClientAliveInterval/CountMax; Windows screen-saver GPO.	Yes — writes sshd_config; reloads sshd; sets registry values.
AT.L2-3.2.1	User awareness training — attestation only.	No — manual.
AU.L2-3.3.1	Audit accountability — auditd enabled and running; rsyslog/journald configured.	Yes — enables auditd, deploys stock rule set, ensures logrotate covers /var/log/audit/.
AU.L2-3.3.3	Audit log review — retrievable logs with >=30 days retention.	Yes — adjusts logrotate retention.
CA.L2-3.12.2	Periodic assessment — attestation.	No — manual.
CM.L2-3.4.1	Baseline configuration — sysctl.conf, limits.conf, sshd_config, Windows security policy vs the CMMC baseline.	Partial — deploys baseline values; flags deltas the operator must accept or override.
CM.L2-3.4.6	Least functionality — running services vs the per-profile allowed list.	Yes (workstation/server). DC profile: manual only.
CM.L2-3.4.2	Security configuration enforcement — presence of CM tooling (ansible, puppet, chef, GPO).	Manual — ForteFide cannot infer the customer's CM approach.
IA.L1-3.5.1	Identifier authentication — PubkeyAuthentication or equivalent; Windows smart card / strong password.	Yes — writes sshd_config; disables password auth after deploying keys.
IA.L2-3.5.7	Multi-factor for privileged access — PAM module presence; Windows MFA registry markers.	No — MFA enrollment is operator-driven.
IA.L2-3.5.10	Password complexity — pam.d/system-auth + pwquality.conf; Windows password policy.	Yes (workstation/server). DC: manual only.
IA.L2-3.5.6	Replay-resistant authentication — SSH cipher review; kerberos-vs-ntlm review.	Workstation/server: yes. DC: manual.

CONTROL	WHAT FORTEFIDE CHECKS	AUTO-REMEDIATES?
IR.L2-3.6.1	Incident response capability — attestation.	No — manual.
MA.L2-3.7.1	Maintenance personnel access — who can log in.	Manual — policy-driven.
MA.L2-3.7.3	Patch level — current package set vs security advisories.	No — patching is operator-driven; ForteFide reports the delta.
MP.L1-3.8.3	Media sanitization — attestation.	No — manual.
MP.L2-3.8.5	Removable media control — udev rules; Windows USB Storage policy registry.	Workstation: yes. Server / DC: probe-only.
PE.L1-3.10.1	Physical access control — attestation.	No — manual.
PS.L2-3.9.2	Personnel screening — attestation.	No — manual.
RA.L2-3.11.2	Risk assessment process — attestation.	No — manual.
SC.L1-3.13.1	Boundary protection — iptables/firewalld presence and rule review; Windows Firewall.	Yes — ensures firewall enabled + default-deny ruleset (with rollback).
SC.L2-3.13.8	Cryptographic protection — SSH cipher suite review; TLS 1.2+ on listening services.	Yes — writes hardened Ciphers/KexAlgorithms/MACs; rollback if validation fails.
SC.L2-3.13.16	Authenticity protection — outbound mail DKIM/SPF; Windows Defender tamper protection.	Partial — enables Defender tamper protection; DKIM/SPF on the mail boundary is manual.
SI.L1-3.14.1	Flaw remediation — package versions vs available security updates.	No — patching is operator-driven.
SI.L2-3.14.3	Security alerts process — security-relevant log forwarding.	Manual — destination depends on the customer's SIEM.
SI.3.220	Tampering integrity — AIDE/Tripwire (Linux); Windows file integrity monitoring registry.	Yes — installs AIDE on Linux (subject to the airgap fallback chain), initializes the database.

Tip — This is a representative sample, not exhaustive — use the CMMC Requirements Guide for the full per-control model.

22 · Operator FAQ

Do I need an internet connection on the scanner?

No. ForteFide is fully self-contained at runtime. License verification, scanning, remediation, and evidence signing all happen offline. The only operations that need internet are buying a license (transferable through the request-code flow) and downloading the airgap .deb (transferable through approved media).

Do I need an internet connection on the targets?

No, with caveats. The scanner reaches targets over your internal network; no target-to-internet access is required. If you remediate controls that need packages the target lacks, ForteFide tries Layer 3 first, then Layer 2 (mounted media) or Layer 1 (vendored bundle).

Can I run ForteFide as a non-root service?

Not currently. The systemd unit runs as root by default (log/data writes, dashboard port binding, target credential handling during prepare). A future version may add a service-user mode.

How do I rotate the per-target SSH cert before 30 days?

Re-run prepare on the target — each prepare issues a fresh 30-day cert. There is no separate "rotate cert" API; the prepare flow IS the rotation flow.

How do I rotate scan_key (the Tier 2 SSH key)?

`POST /api/generate-scan-keys` regenerates `DATA_DIR/keys/scan_key`. Re-run prepare on each target to deploy the new public key.

What happens if a remediation makes a target unreachable?

If the in-OS watchdog is armed and the heartbeat stays stale beyond the watchdog window, the target self-heals from its reversion bundle. If no watchdog and no hypervisor snapshot were taken, the operator must manually recover the target. The journal entry stays open until the target is reachable again.

Can I use ForteFide to scan cloud workloads (AWS, Azure, GCP)?

Yes, as long as you have network reachability over SSH or WinRM. ForteFide does not integrate with cloud control planes — it treats cloud VMs like any other Linux/Windows host.

How do I integrate ForteFide with my SIEM?

`GET /api/logs/export` returns CSV; alternatively, the audit log is RFC-3164-compatible and can be tail-shipped via rsyslog / filebeat / a Splunk forwarder. ForteFide produces logs and lets your existing infrastructure consume them.

My license shows 24/25 seats but I have 25 hosts — what happens at host 26?

A seat is a currently-prepared endpoint. Preparing the 26th distinct host is rejected with HTTP 403 and a hint to upgrade or tear down a prepared host first. Scanning the 26th host still works — only prepare and remediation are gated.

Does ForteFide work on Active Directory members?

Yes. AD members and Domain Controllers are both supported. The DC risk profile filters IA.L2-3.5.6 and CM.L2-3.4.6 to avoid mid-batch self-lockout. ForteFide creates a LOCAL fortefide-svc account on each target, not a domain account.

Can I export evidence in a format other than ZIP?

`GET /api/scan/<id>/json | csv | pdf` return individual formats. The signed evidence package is always a ZIP because the signature covers the manifest, which lists the files.

How long should I keep signed evidence packages?

CMMC retention guidance applies (currently 6 years). Store the ZIP and .sig together; signatures verify offline with the published DenseDefense public key, and remain valid even after the producing license expires.

What if DenseDefense disappears — can I still verify old evidence?

Yes. The DenseDefense public key is a static Ed25519 key that customers should archive alongside their evidence retention. Any openssl/ssh-keygen-equipped reader can verify the manifest signature with the archived public key indefinitely.

23 · Appendix A: REST API Surface

ForteFide exposes a large REST API on the dashboard port. The lists below group routes by feature area; consult the API Reference document for per-route schemas, error codes, and examples.

Scan & discovery

```
POST /api/scan                          # start a scan
GET  /api/scans                         # list past scans
GET  /api/scan/<id>                     # full scan record
POST /api/scan/<id>/abort              # graceful abort
GET  /api/scan/<id>/delta/<prev_id>    # delta vs prior scan
GET  /api/scan/history                  # compliance over time
GET  /api/scan/<id>/json|csv|pdf        # exports
POST /api/discover                      # CIDR / list discovery
GET  /api/discover/<id>                 # discovery status
POST /api/discover/<id>/abort           # graceful abort
GET  /api/inventory                     # host inventory
GET  /api/matrix                        # control x host matrix
GET  /api/matrix/<control_id>           # one control across hosts
```

Remediation & rollback

```
POST /api/remediate                     # one control on one host
POST /api/remediate-batch               # all controls on one host
POST /api/remediate-fleet               # one control across fleet
GET  /api/remediate-fleet/<job_id>      # job status
POST /api/remediate-fleet/<job_id>/abort
POST /api/remediate-async               # background job
GET  /api/remediate-job/<job_id>        # async job status
POST /api/remediate-job/<job_id>/abort
POST /api/rollback                      # one entry
POST /api/rollback-batch                # one target, all entries
POST /api/rollback-all                  # fleet-wide rollback
GET  /api/journal/status                # open / reversed / failed
GET  /api/reversion-bundles             # bundle list
GET  /api/reversion-bundles/<host>/<ts>/download
GET  /api/remediation-history           # rebuilt-from-journal
GET/POST /api/settings/remediation-workers   # parallel workers
```

Endpoint preparation, watchdog, CAR

```
POST /api/prepare-endpoint               # phase 0/0+/1/2
GET  /api/prepare-endpoint/<prep_id>     # prep status
POST /api/teardown-endpoint               # leave-no-trace
GET  /api/svc-account/<target>          # one host's svc account
GET  /api/svc-accounts                    # all prepared accounts
GET/PUT /api/target-profile/<target>     # risk profile
GET  /api/target-profiles                 # all profiles
GET  /api/watchdog                        # all targets
GET  /api/watchdog/<target>              # one target
POST /api/watchdog/<target>/disarm       # planned outage
GET  /api/customer-action                 # all pending CAR
GET  /api/customer-action/<target>       # one target's CAR
POST /api/customer-action/<target>/retry
```

Evidence

```
GET  /api/scan/<id>/evidence-package          # draft (free)
POST /api/scan/<id>/signed-evidence-package   # signed (licensed)
GET  /api/scan/<id>/evidence/<doc_type>
GET  /api/evidence-status                     # invalidated by rollback?
```

License, scheduler, definitions, control overrides

```
GET  /api/request-code               # FIDE-XXXX-XXXX-XXXX
POST /api/import-license             # drag-drop license.key
POST /api/activate                   # legacy activate flow
GET  /api/license-status             # tier, expiry, days
POST /api/control-override           # mark control as N/A
GET  /api/control-overrides
DELETE /api/control-override/<key>
GET  /api/scheduler/status           # Beta
GET/POST /api/scheduler/config       # Beta
POST /api/scheduler/start            # Beta
POST /api/scheduler/stop             # Beta
GET  /api/scheduler/history          # Beta
GET  /api/definitions/status         # bundle freshness
POST /api/definitions/import         # offline def import
GET  /api/definitions/history
POST /api/definitions/dismiss-reminder
```

Resource groups & operational

```
GET  /api/resource-groups
POST /api/resource-group                    # create
DELETE /api/resource-group/<name>
POST /api/resource-group/<name>/hosts       # add hosts
GET  /api/health
GET  /api/capabilities                      # build flags, tier
GET  /api/logs                              # in-memory ring
GET  /api/logs/export                       # CSV
GET  /api/session-state
POST /api/session-clear
POST /api/generate-scan-keys                # regenerate scan_key
GET  /api/scan-keys-status
POST /api/restart                           # graceful restart
```

24 · Appendix B: Files & Directories on the Scanner

PATH (LINUX; WINDOWS = %PROGRAMDATA%\FORTEFIDE\)	PURPOSE	TOUCH ON UNINSTALL?
/opt/fortefide/fortefide	ELF binary	Yes — removed.
/opt/fortefide/templates/dashboard.html	Single-page dashboard	Yes — removed.
/opt/fortefide/static/	JS, CSS, images	Yes — removed.
/opt/fortefide/packages/<family>/<version>/	Layer 1 vendored bundles (airgap variant)	Yes — removed.
/opt/fortefide/keys/fortefide_public.key	Ed25519 bundle-verification public key	Yes — removed.
/var/lib/fortefide/license.key	Imported license; back up before reinstall	Preserved on dpkg -i; WIPED on dpkg --purge.
/var/lib/fortefide/scan.db	Encrypted SQLite scan history	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/journal/remediation.jsonl	Durable remediation journal	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/certs/<target>/	Per-target SSH cert + key + metadata	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/keys/scan_key	Long-lived Ed25519 SSH key (Tier 2)	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/target_profiles.json	Per-target risk profile overrides	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/_svc_accounts.json	Prepared service-account registry	Preserved on dpkg -i; wiped on --purge.
/var/lib/fortefide/.eula_accepted	Marker file (EULA acceptance timestamp)	Preserved on dpkg -i; wiped on --purge.
/var/log/fortefide/fortefide.log	Append-only audit log	Not removed by uninstall; logrotate handles rotation.
/etc/systemd/system/fortefide.service	Systemd unit	Yes — removed and disabled.

25 · Appendix C: Files & Directories on a Prepared Target

What ForteFide creates on a Linux target during prepare. Teardown removes all of these per the per-target reversion bundle — ForteFide does not leave anything behind that is not in this list.

ARTIFACT	PATH (LINUX)	PATH (WINDOWS)
Service account	User fortefide-svc; home /home/fortefide-svc	User fortefide-svc; profile %SystemDrive%\Users\fortefide-svc
Sudoers entry	/etc/sudoers.d/fortefide-svc	Administrators group membership
Bootstrap-user NOPASSWD (if added by Phase 0)	/etc/sudoers.d/<user>-nopasswd	n/a
scan_key public key	~fortefide-svc/.ssh/authorized_keys (entry)	%ProgramData%\ssh\administrators_authorized_keys (entry)
30-day cert public key	~fortefide-svc/.ssh/authorized_keys (entry)	%ProgramData%\ssh\administrators_authorized_keys (entry)
CA root public key	/etc/ssh/fortefide_ca.pub	n/a (paramiko fallback path on Windows — not deployed)
sshd_config TrustedUserCAKeys append	Line in /etc/ssh/sshd_config	n/a
Watchdog cron / scheduled task	Cron entry under fortefide-svc	Scheduled task "ForteFide Watchdog"
Reversion bundle	/var/lib/fortefide/reversion-bundle.json	%ProgramData%\ForteFide\reversion-bundle.json

Tip — After Teardown, the target should have: zero ForteFide-named users, zero ForteFide-named files in /etc/sudoers.d, no ForteFide entries in any authorized_keys, no /etc/ssh/fortefide_ca.pub, no TrustedUserCAKeys line referencing it, no fortefide_* cron entries, and no scheduled tasks named ForteFide. The signed evidence package is the artifact of record — the customer's machine is restored.

26 · Appendix D: Glossary

TERM	DEFINITION
Auth cascade	ForteFide's three-tier authentication preference: cert (Tier 1) → ssh_key (Tier 2) → password (Tier 3). Higher tier preferred; lower tier as fallback.
Auto-teardown trap	Historical behavior where any rescan after _remediation_history was non-empty wiped svc accounts for the whole fleet. Disabled in the current product — Teardown is operator-triggered only.
CA (in this product)	Certificate Authority. ForteFide derives an SSH CA keypair from the license HKDF seed; the CA private key is wiped from memory after each prepare; the CA public key is the trust root deployed to targets.
CAR	Customer Action Required. The signal ForteFide raises when a specific operation cannot proceed without operator intervention — typically an unavailable package on an air-gapped target. Narrow by design.
C3PAO	CMMC Third-Party Assessment Organization — the accredited assessor who reviews the customer's CMMC evidence. ForteFide produces signed evidence packages for C3PAO consumption.
DATA_DIR	ForteFide's data directory. Linux: /var/lib/fortefide. Windows: %ProgramData%\ForteFide. Holds license, scan DB, journal, certs, profiles, and svc-account registry.
DC profile	Risk profile applied to Windows Domain Controllers. Auto-skips IA.L2-3.5.6 + CM.L2-3.4.6 to prevent mid-batch self-lockout from domain policy changes.
Evidence package	ZIP of scan results + per-control evidence + host inventory + remediation history. Draft (free) is unsigned; Signed (licensed) is Ed25519-signed and bound to the scanner's machine fingerprint.
fortefide-svc	The local service account ForteFide creates on each target during prepare — the only account ForteFide uses post-prepare.
Free tier	Scanner-only mode. No license required. Scan and discover work; remediate, prepare, and signed evidence are gated.
HKDF	HMAC-based Key Derivation Function — used to derive the SSH CA seed from the license payload.
Journal	Append-only JSONL of every successful remediation. Drives rollback; replayed at startup so rollback survives service restarts.
Layer 1 / 2 / 3	Airgap install layers. Layer 3 = target's own repos (default). Layer 2 = customer-mounted install ISO. Layer 1 = ForteFide-vendored bundle. Fallback priority: 3 → 2 → 1 → CAR.
Leave-no-trace	ForteFide's design doctrine: every state change is recorded in a per-target reversion bundle; teardown undoes those changes against the bundle. Post-engagement the customer's machine is restored; the signed evidence package is the only retained artifact.
NOPASSWD	Sudoers attribute meaning "this user can sudo without supplying a password" — required on the bootstrap user for Prepare to function; Phase 0 grants this if missing.

TERM	DEFINITION
Phase 0 / 0+ / 1 / 2	Prepare's four phases: 0 = NOPASSWD sudo; 0+ = baseline tools; 1 = service account creation; 2 = SSH key + cert deployment.
Recon	Step 1 of the customer journey — discovers reachable hosts on a CIDR, guesses OS, produces the prepare candidate list.
Reversion bundle	Per-target JSON file with everything ForteFide changed and the data needed to undo it. Lives on the target; read by teardown; never assumed beyond what it records.
scan_key	The long-lived Ed25519 SSH key ForteFide deploys for Tier 2 fallback — one per scanner, reused across all targets.
Seat	One currently-prepared endpoint (held in _svc_accounts). Preparing a host consumes a seat; tearing it down frees the seat. Scanning does NOT consume a seat.
Signed evidence	Evidence package with Ed25519 signature over the manifest plus license fingerprint — verifiable offline by the C3PAO assessor with the published DenseDefense public key.
Tier 1 / 2 / 3 (auth)	ForteFide's customer-facing auth strength numbering: Tier 1 = strongest (cert), Tier 3 = weakest (password).
TrustedUserCAKeys	OpenSSH server config directive that tells sshd which CA public keys are trusted to sign user certs. ForteFide adds /etc/ssh/fortefide_ca.pub on Linux.
Watchdog	In-OS heartbeat-driven safety net deployed during prepare — self-heals the target if remediation stalls without recording completion.

ForteFide v26.07.27.1120 — CMMC(TM) Level 2 / NIST SP 800-171 Rev 2 compliance-readiness platform. ForteFide helps organizations prepare for a CMMC assessment; it does not perform assessments, certify compliance, or replace a C3PAO.