API REFERENCE V26.07.27.1120 DENSEDEFENSE · FORTEFIDE

ForteFide API Reference

Complete REST reference for the ForteFide product API (app_source.py, port 5000) and the vendor-side Admin API (admin_api.py, port 9410) -- request/response shapes, error codes, and license gates, generated from the live route table.

- 79 of 110 controls scannable · 31 manual-only
- Product API · token auth (X-Auth-Token) · loopback by default
- Admin API · Bearer + HMAC, never shipped to customers

Base URLs & conventions

PRODUCT API

`http://<host>:5000` (or `FORTEFIDE_BIND_PORT` override) -- runs on the customer's scanner host, JSON in/out, binds to `127.0.0.1` by default; every route requires the access token (`X-Auth-Token` header)

ADMIN API

`http://<host>:9410` -- vendor-side license provisioning; Bearer `ADMIN_API_KEY` except the HMAC-signed payment webhook; never compiled into a customer-facing build

ERRORS

Common shape: `{"error": "...", "hint": "...", "code": "...", "upgrade": true}` -- `upgrade: true` appears on Pro-gated 403s

CONTENTS

1 · Overview

The ForteFide REST API provides programmatic access to compliance scanning, discovery, remediation, rollback, evidence packaging, licensing, and fleet management. Two HTTP services are documented here:

- **Product API** (`app_source.py`): port 5000 by default. Runs on the customer's scanner host. Binds to `127.0.0.1` (loopback) by default; every route requires the access token as an `X-Auth-Token` header (or the browser session cookie). Retrieve the token on the scanner host with `fortefide --show-token`.
- **Admin API** (`admin_api.py`): port 9410. Vendor-side license provisioning service. Bearer-token authenticated; the payment webhook uses HMAC. Never compiled into the customer-facing build.

Conventions (Product API)

- Base URL: `http://<host>:5000` (or `FORTEFIDE_BIND_PORT` override); binds to `127.0.0.1` by default.
- **Authentication:** every `/api/*` route requires the access token as an `X-Auth-Token` header (or the `ff_dashboard_auth` session cookie a browser gets after signing in). Without it, routes return `401`. Retrieve the token on the scanner host with `fortefide --show-token`. Example: `curl -H "X-Auth-Token: $(fortefide --show-token)" http://127.0.0.1:5000/api/health`
- Request bodies are JSON (`Content-Type: application/json`); multipart accepted on `/api/definitions/import` and `/api/attestation/<family>`.
- Responses are JSON unless noted (PDF/CSV/JSON downloads, ZIP evidence packages, tarball reversion bundles).
- All responses include `Cache-Control: no-cache, no-store, must-revalidate`. CORS is open.
- IDs (scan, discovery, prep, remediation job) are sequential integers.
- Completed scans persist to AES-256-GCM encrypted SQLite and auto-restore on restart. In-memory state (discoveries, preparations, remediation jobs, svc accounts) is cleared on restart.

Common error envelope

Most error responses use a consistent shape with optional fields:

```
{
  "error": "Short customer-facing summary.",
  "hint": "Actionable next step (when available).",
  "code": "machine_readable_error_code",
  "upgrade": true  // included on Pro-gated 403s
}
```

Status codes used by the product API: 200 (success), 400 (bad request / validation), 403 (license-gated or forbidden), 404 (not found), 409 (conflict / refuse-while-busy), 500 (server error), 503 (feature not yet wired).

Tip — The catalogue currently covers 79 of 110 NIST SP 800-171 controls with an automated scan check (31 are manual-only by nature); of those, 82 have a safe auto-remediation across Linux and/or Windows (67 Linux, 66 Windows). These counts are re-derived live at `GET /api/engagement/pregame` -- never hand-typed into a report.

2 · Endpoint summary

Every route below is grepped live from the running app -- not copied from a prior doc pass. Pro markers indicate endpoints that require a valid license (`REMEDIATION_AVAILABLE` or `SCHEDULER_AVAILABLE`).

System

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/health`	Liveness probe; uptime + version
GET	`/api/capabilities`	Feature flags for the running instance
GET	`/api/inventory`	Prepared svc accounts + remediation key stats
GET	`/api/logs`	Activity log entries (`?since=N` for delta polling)
GET	`/api/logs/export`	Download activity log as JSON or CSV
GET	`/api/failure-report`	Aggregated failure-classification report (json|csv)
POST	`/api/restart`	Restart server (refuses on in-flight work)
GET	`/api/request-code`	Machine-bound request code for licensing
GET	`/api/session-state`	Lean state summary for UI restore
POST	`/api/session-clear`	Destructive: wipe all session state

Scanning

METHOD	ENDPOINT	DESCRIPTION
POST	`/api/scan`	Start a compliance scan
GET	`/api/scan/{id}`	Get scan state and results
POST	`/api/scan/{id}/abort`	Abort a running scan
GET	`/api/scans`	List scans (lean summaries by default)
GET	`/api/scan/history`	Score history for trend graphs
GET	`/api/scan/{id}/delta/{prev_id}`	Compare two completed scans
GET	`/api/scan/{id}/json`	Export scan as downloadable JSON
GET	`/api/scan/{id}/csv`	Export findings or all controls as CSV
GET	`/api/scan/{id}/pdf`	Export scan report as PDF
GET	`/api/scan/{id}/basic-report`	Free compliance summary PDF (no remediation steps, no signed evidence)
POST	`/api/scan/{id}/signed-evidence-package`	Pro: signed evidence ZIP
GET	`/api/scan/{id}/evidence-package`	Unsigned evidence ZIP (free/DRAFT)
GET	`/api/scan/{id}/evidence/{doc_type}`	Single evidence doc download
POST	`/api/scan/{id}/vault-upload`	DDWitness: push encrypted signed evidence to vault

Discovery

METHOD	ENDPOINT	DESCRIPTION
POST	`/api/discover`	Start network/AD/LDAP/proxy discovery
GET	`/api/discover/{id}`	Get discovery state and results
POST	`/api/discover/{id}/abort`	Stop a running discovery

CMMC matrix

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/matrix`	Full CMMC/NIST 800-171 control matrix (filterable client-side)
GET	`/api/matrix/{control_id}`	Single control entry

Remediation & rollback -- Pro

METHOD	ENDPOINT	DESCRIPTION
POST	`/api/remediate`	Remediate single control on one target
POST	`/api/remediate-batch`	Batch-remediate controls on one target
POST	`/api/remediate-fleet`	Parallel remediate across many targets
GET	`/api/remediate-fleet/{id}`	Poll fleet job progress
POST	`/api/remediate-fleet/{id}/abort`	Abort fleet job
POST	`/api/remediate-async`	Async batch remediate (returns job_id)
GET	`/api/remediate-job/{id}`	Poll async job progress
POST	`/api/remediate-job/{id}/abort`	Abort async job
POST	`/api/remediate-engagement-plan`	Deterministic remediation plan derived from a scan_id
GET	`/api/remediation-history`	Session + journal history (CSV/JSON)
GET POST	`/api/settings/remediation-workers`	Get/set fleet max workers
POST	`/api/rollback`	Undo single remediation
POST	`/api/rollback-batch`	Undo multiple on one target
POST	`/api/rollback-all`	Durable Revert-All from journal
GET	`/api/journal/status`	Open journal entries summary
POST	`/api/journal/abandon`	Mark journal entries abandoned (uninstall flow)
GET	`/api/reversion-bundles`	List local reversion bundles
GET	`/api/reversion-bundles/{host}/{ts}/download`	Download bundle tarball

Endpoint preparation -- Pro

METHOD	ENDPOINT	DESCRIPTION
POST	`/api/preflight-credentials`	Validate admin creds authenticate before Prepare mutates anything
POST	`/api/prepare-endpoint`	Create svc account on target
GET	`/api/prepare-endpoint/{id}`	Poll prep status (password redacted)
POST	`/api/teardown-endpoint`	Remove svc account; auto-rollback first
POST	`/api/teardown-batch`	Parallel multi-target teardown
GET	`/api/svc-account/{target}`	Stored svc creds for one target
GET	`/api/svc-accounts`	List prepared endpoints (with is_dc/tearable)
GET	`/api/target-profile/{target}`	Get risk profile for target
PUT POST	`/api/target-profile/{target}`	Set risk profile
GET	`/api/target-profiles`	List all per-target profiles
GET	`/api/watchdog`	Watchdog state (all targets)
GET	`/api/watchdog/{target}`	Watchdog state for one target
POST	`/api/watchdog/{target}/arm`	Manually arm watchdog out-of-band
POST	`/api/watchdog/{target}/disarm`	Manually disarm watchdog (maintenance window)
POST	`/api/watchdog/{target}/soak`	Set watchdog soak window (seconds) for planned downtime
GET POST	`/api/firewall-ack`	List / record signed operator acknowledgment for firewall-doctrine controls
GET	`/api/customer-action`	List targets with pending Customer-Action-Required prompts
GET	`/api/customer-action/{target}`	CAR detail for one target
POST	`/api/customer-action/{target}/retry`	Retry prepare after customer mounts install media

Resource groups & control overrides

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/resource-groups`	List groups (passwords redacted)
POST	`/api/resource-group`	Create or update a group
POST	`/api/resource-group/{name}/hosts`	Add or remove hosts
POST	`/api/control-override`	Mark control as exception or alt-implementation
GET	`/api/control-overrides`	List all overrides

Licensing

METHOD	ENDPOINT	DESCRIPTION
POST	`/api/import-license`	Import license.key file
POST	`/api/activate`	Activate Pro with one-time activation code
POST	`/api/claim-license`	Exchange a claim token for a signed, machine-bound license
GET	`/api/license-status`	License tier, expiry, request code
GET	`/api/evidence-status`	Evidence validity (rollback-invalidated?)

Scan scheduler & definition updates -- Pro / Beta

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/scheduler/status`	Scheduler + definitions status
GET POST	`/api/scheduler/config`	Get / update scheduler config
POST	`/api/scheduler/start`	Start scheduler thread (Beta -- see §12)
POST	`/api/scheduler/stop`	Stop scheduler thread
GET	`/api/scheduler/history`	Scheduled-run history
GET	`/api/definitions/status`	Current definition version + age
POST	`/api/definitions/import`	Import signed definition package
GET	`/api/definitions/history`	Definition update audit history
POST	`/api/definitions/dismiss-reminder`	Suppress staleness reminder

Expansion bundles & DDWitness

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/expansions/installed`	Layer 1 inventory of /opt/fortefide/packages/
GET	`/api/expansions/required`	Cross-reference detected fleet vs Layer 1; emit download recommendations
POST	`/api/expansions/install`	Multipart .tar.xz + Ed25519 .sig upload; verify + extract
GET	`/api/vault/status`	Is FF configured to push to DDWitness vault?
POST	`/api/vault/configure`	Set vault upload token (Pro-only)
POST	`/api/vault/clear`	Remove configured token

Engagement lifecycle

METHOD	ENDPOINT	DESCRIPTION
GET	`/api/engagement/pregame`	Pre-engagement coverage report to acknowledge before Prepare/Scan
POST	`/api/engagement/pregame/acknowledge`	Record the operator's acknowledgment
GET	`/api/engagement/{id}/status`	Single-poll phase signal (baseline/remediation running/complete)
GET	`/api/engagement/{id}/postmortem`	Per-target per-control failure analysis for a completed engagement
GET	`/api/state-changes` · `/{engagement_id}`	HKDF/HMAC-signed install/remove records (leave-no-trace verification)
POST	`/api/attestation/{family}`	Upload signed attestation PDF for a manual CMMC family
GET	`/api/attestation/status`	Attestation upload status for all manual families

Operator settings, outreach, warranty, scan keys

METHOD	ENDPOINT	DESCRIPTION
GET POST DELETE	`/api/settings/excluded-targets`	Persistent IP/hostname out-of-scope list; discovery filters every engagement
POST	`/api/request-call`	Self-serve customer outreach; persists to call_requests.jsonl
GET	`/api/request-call/log`	Operator-side read of pending outreach
GET	`/api/warranty-status`	Current warranty-check state
GET POST	`/api/warranty-settings`	Read / toggle online warranty-check opt-in
POST	`/api/warranty-check`	Trigger an immediate warranty check (no-op if opt-out)
POST	`/api/generate-scan-keys`	Generate SSH+cert pair (refuses if endpoints prepared)
GET	`/api/scan-keys-status`	Check if keys/certs exist on disk

Admin API (admin_api.py, port 9410)

METHOD	ENDPOINT	DESCRIPTION
OPTIONS	`/api/<path>`	CORS preflight
POST	`/api/provision`	Provision a single license
POST	`/api/provision/batch`	Batch provisioning from JSON array
POST	`/api/webhook/payment`	HMAC-signed payment webhook
GET	`/api/license/{org_id}/download`	Download provisioned license.key
POST	`/api/license/{org_id}/renew`	Renew an existing license
GET	`/api/provision/log`	Query provision audit trail

UI / misc

METHOD	ENDPOINT	DESCRIPTION
GET	`/`	Serve dashboard.html

3 · System

`GET` `/api/health`

Liveness probe. Returns 200 with status, version, and uptime in seconds.

```
{"status": "ok", "version": "26.07.27.1120", "uptime_seconds": 3600}
```

`GET` `/api/capabilities`

Feature flags for the running instance. Call this before hitting Pro-gated endpoints; it re-evaluates cert/key paths on every call so `/api/generate-scan-keys` results are visible without a restart.

`GET` `/api/logs`

Activity log entries from the 3000-entry circular buffer. Pass `?since=N` to fetch only entries with `id > N` (efficient delta polling). Categories: SYSTEM, SCAN, PROGRESS, REMEDIATE, ROLLBACK, DISCOVERY, PREPARE, EVIDENCE, LICENSE, GROUPS, OVERRIDE, SCHEDULER, SETTINGS, CONFIG.

`POST` `/api/restart`

Restart the server process. Refuses while scans, preparations, or remediation jobs are in flight (409). Pass `{"force": true}` to override.

> **Note —** 409: `scan_in_progress` / `prepare_in_progress` / `remediate_in_progress` -- pass `force: true` to interrupt anyway.

`POST` `/api/session-clear`

Destructive: wipes all scans, svc accounts, remediation history, control overrides, resource groups, and the activity log.

```
{"confirm": "CLEAR_ALL"}  // mandatory phrase
{"confirm": "CLEAR_ALL", "force_orphan": true}  // override prepared-endpoints check
```

High-impact — Refuses if any svc accounts are still prepared (409 `prepared_endpoints_exist`) -- override with `force_orphan`. Source IP + user-agent are logged before clearing so the event persists.

`GET` `/api/failure-report`

Aggregated failure report across every durable source ForteFide writes. Classifies each failure into one of 11 categories (timeout, ssh-fail, auth-fail, sudo-fail, command-not-found, network, layer1-missing, exit-code, permission, watchdog-deploy, license). Query params: `since=YYYY-MM-DD`, `format=json|csv`.

4 · Scanning

`POST` `/api/scan`

Start a compliance scan against one or more targets. Two body shapes accepted: comma-separated string or an array of per-target objects. Quick-scan ignores authentication; full scan requires creds (or a prepared svc account on the target).

```
{
  "targets": "198.51.100.10,198.51.100.11",
  "mode": "full",
  "username": "fortefide-svc",
  "password": "<password>",
  "os_type": "windows",
  "auth_mode": "password"
}
```

FIELD	TYPE	REQUIRED	DEFAULT	DESCRIPTION
`targets`	string|array	Yes	--	Comma IPs or target objects
`mode`	string	No	`quick`	`quick` (port probe) or `full` (79 of 110 controls, varies by target OS)
`username` / `password`	string	For full	--	WinRM/SSH credentials
`os_type`	string	No	`windows`	`windows` / `linux` / `auto`
`auth_mode`	string	No	`password`	`password` / `certificate` / `inventory`
`proxy_host` / `proxy_port` / `proxy_username` / `proxy_password`	--	If jump-host	--	Bastion for a Linux-only jump-host scan
`credential_profiles`	array	No	--	Optional per-target credential mappings

```
200: {"scan_id": 1, "status": "queued", "mode": "full"}
```

Note — 400: `bad_target` / `bad_os_type` / `bad_control_id` / `bad_targets_type` / `too_many_targets` (cap 500 per call) / `bad_proxy_port`.

`GET` `/api/scan/{scan_id}`

Scan state and full results. While running, returns lightweight progress fields only (id, status, progress, started, elapsed_sec, targets, mode, os_type). Once complete, returns the full record; falls through to the encrypted scan store for older scans.

`GET` `/api/scans`

List scans newest-first. Lean summaries (id, status, score, host count) by default to keep the dashboard refresh small even with 1000+ scans. `?verbose=1` for full records, `?paginated=1` to wrap in `{total, offset, limit, count, scans}`.

Reports & evidence

- `/api/scan/{id}/pdf` -- formatted PDF scan report.
- `/api/scan/{id}/basic-report` -- free compliance summary PDF (both editions); carries no remediation steps or signed evidence package.
- `/api/scan/{id}/evidence-package` -- unsigned DRAFT evidence ZIP (free tier): SSP, POA&M, asset inventory, SPRS, control evidence, remediation log, IRP, training records, policies, manifest.
- `/api/scan/{id}/signed-evidence-package` (POST, Pro) -- cryptographically signed evidence ZIP with Ed25519 chain-of-custody.

Note — 409: evidence `INVALIDATED` if a rollback occurred since the last scan -- re-scan before generating evidence.

5 · Discovery

`POST` `/api/discover`

Start host discovery using one of four methods: network scan, Active Directory LDAP query, raw LDAP, or proxy/jump-host discovery. Unknown methods are rejected at the edge with a hint.

```
// Network
{"method": "network", "cidr": "10.0.0.0/24"}

// Active Directory
{"method": "ad", "dc_ip": "10.0.0.10", "domain": "corp.example.com",
 "dc_username": "Administrator", "dc_password": "<pw>"}

// Proxy / jump-host
{"method": "proxy", "proxy_host": "198.51.100.10", "proxy_port": 22,
 "proxy_username": "jumpuser", "proxy_password": "<pw>",
 "proxy_target": "network", "cidr": "203.0.113.0/24"}
```

Note — 400: `bad_method` (valid: network, ad, ldap, proxy).

`GET` `/api/discover/{disc_id}`

Poll discovery state. Returns a `hosts` array (ip, hostname, os, source, ports) and `stats` (total/windows/linux/unknown host counts).

6 · CMMC control matrix

`GET` `/api/matrix`

Returns the full CMMC / NIST SP 800-171 control matrix (110 controls). Automation/remediation commands are **stripped** before serialization -- never exposed over HTTP. No query parameters: the endpoint always returns the full set; filter client-side.

```
{
  "AC.L1-3.1.1": {
    "title": "Limit information system access...",
    "family": "AC", "severity": "high",
    "description": "...", "remediation_steps": [...],
    "nist_ref": "3.1.1", "references": []
  }
}
```

`GET` `/api/matrix/{control_id}`

Single control entry. Same fields as the matrix above.

> **Note —** 404: control not found.

7 · Remediation (Pro)

All remediation endpoints require a valid license. Unlicensed requests return 403: `{"error": "Remediation requires a valid license. Import one via /api/import-license (the scanner stays free).", "upgrade": true}`.

`POST` `/api/remediate`

Execute automated remediation for a single control. Resolves credentials from the prepared svc account if username/password are not supplied. Catalogue commands are decrypted in memory and executed via WinRM/SSH/PSRP.

```
{"control_id": "AC.L1-3.1.1", "target": "10.0.1.50",
 "username": "fortefide-svc", "password": "<pw>", "os_type": "windows"}
```

`POST` `/api/remediate-fleet`

Parallel remediation across many targets via ThreadPoolExecutor. Per host: groups controls by family and runs families in parallel (max 4 concurrent; controls within a family run sequentially). `max_workers` capped at 20; fleet size capped at 500 per call; seat limit enforced against the license's `max_endpoints`.

```
200: {"job_id": 5, "status": "running", "total_hosts": 2,
 "total_controls": 4, "max_workers": 8}
```

`POST` `/api/remediate-async`

Start an async batch remediation job for one target; poll `/api/remediate-job/{id}`. Supports per-host credential profiles, automatic `os_type` detection (port probe), pre-flight checks, and DD-Trust Layer 3 post-state verification.

`POST` `/api/remediate-engagement-plan`

Server-derived deterministic remediation plan from a scan record. Same `(scan_id, filters)` input always produces the same plan -- hosts and control IDs are sorted; the plan filters to prepared hosts only, and drops MANUAL / not-in-catalogue controls when `auto_remediable_only` is true.

```
{"scan_id": 42, "severities": ["critical","high","medium","low"],
 "auto_remediable_only": true, "targets": null}
```

Caller workflow: POST `scan_id`, receive the plan, iterate it calling `/api/remediate-async` per host with that host's `control_ids`.

`GET` `/api/remediation-history`

Session + journal-restored remediation history. Each entry is tagged with `rollback_available`.

8 · Rollback (Pro)

Rollback reverses previously applied remediations. Tracking lives in the durable on-disk JSONL journal plus in-memory history. Of the 82 controls with a safe auto-remediation, the majority have rollback commands; a small set are intentionally one-way and are reported separately on rollback failure.

`POST` `/api/rollback`

Undo a single remediation. Validates the control was remediated on this target and runs a pre-flight reachability probe before attempting reverse-commands.

High-impact — 503: target unreachable on standard management ports (TCP/22 SSH, TCP/5985 WinRM) -- rollback cannot be sent over the same channel that's down. Response includes `code=target_unreachable` and points the operator at `/var/lib/fortefide/emergency-revert.sh` on the target's console.

`POST` `/api/rollback-all`

Durable Revert-All from the on-disk journal. Reads open entries, groups by target, executes batch rollback per target using prepared svc creds (or supplied admin creds). Survives a service restart. Pass `{"dry_run": true}` to preview the plan without executing.

Tip — All three rollback endpoints probe the target's management ports (2-second timeout) BEFORE attempting reverse-commands, so a broken-connectivity host fails fast with `target_unreachable` instead of hanging.

`GET` `/api/journal/status`

Observability for the durable journal: `journal_path`, `open_entry_count`, and a per-target list of pending `control_ids`.

`GET` `/api/reversion-bundles/{hostname}/{timestamp}/download`

Download a reversion bundle tarball. `hostname` / `timestamp` are validated against strict charsets plus a realpath containment check to block path traversal.

9 · Endpoint preparation (Pro)

Endpoint preparation creates a dedicated service account on a target, deploys SSH keys/certs, and registers the target in the in-memory inventory. Teardown removes SSH access (key/cert + sshd_config edits) but leaves the local user inert -- no admin credential material persists in scanner state beyond the prep step.

Transport options

Both prepare and teardown accept an optional `transport` field for Windows targets:

VALUE	PORT	DESCRIPTION
`auto`	22/5985	Default. Probes ports, prefers SSH > PSRP > WinRM
`winrm`	5985	WinRM with NTLM (pywinrm), ~8 KB encoded-command limit
`ssh`	22	OpenSSH Server with `-EncodedCommand` (paramiko)
`psrp`	5985	PowerShell Remoting Protocol (pypsrp)

Linux targets always use SSH.

`POST` `/api/preflight-credentials`

License-gated. Validates that supplied admin credentials authenticate to each target BEFORE Prepare mutates anything -- auth-only, no target is altered. Provide Windows creds (`win_username` / `win_password`) and/or Linux creds (`lin_username` / `lin_password`); at least one set is required. Max 500 targets.

`POST` `/api/prepare-endpoint`

License-gated. Creates a `fortefide-svc` account on the target, generates a per-target SSH key + signed cert, deploys the public key/cert + CA, captures pre-state, and arms the in-OS watchdog (Linux). Async -- poll `/api/prepare-endpoint/{id}` for status.

`POST` `/api/teardown-endpoint`

Removes the deployed SSH key/cert + sshd_config trust + firewall rule. Auto-rolls-back pending journal entries first. Failures are classified as true failures (409, blocks teardown) vs intentional one-ways (lets teardown proceed). Username/password in the request body are IGNORED -- teardown uses the deployed key/cert.

`POST` `/api/teardown-batch`

Tears down multiple prepared endpoints in parallel (ThreadPoolExecutor, max_workers=4). Per-target failure isolation; targets deduped before dispatch; hard cap 256 targets per request.

Watchdog

`GET /api/watchdog` / `/{target}` return armed_at, deploy_result, pre_snapshot, heartbeat_log, rollback_fired. `POST /{target}/arm` lets an operator re-arm out-of-band after fixing a transient SSH issue. `POST /{target}/soak` sets the soak window (seconds, bounded [60, 86400]) for planned maintenance -- `seconds=0` disarms.

Firewall remediation: signed operator acknowledgment

Firewall remediations (AC.L1-3.1.20, SC.L1-3.13.1, SC.L2-3.13.6, SC.L1-3.13.5 -- same IDs on Linux and Windows) are MANUAL by default. ForteFide refuses to auto-apply any of these until the operator has produced a signed acknowledgment for (target, control_id) via `POST /api/firewall-ack`. Acknowledgments are HMAC-SHA256-signed with the license-derived state-change key, persist to `journal/firewall_acks.jsonl`, and bundle into the signed evidence ZIP.

High-impact — Firewall changes are the highest-blast-radius remediations ForteFide runs -- a misapplied rule can lock out the operator (covered by watchdog) AND every dependent customer process (not covered by watchdog). The signed acknowledgment is the doctrine gate.

Customer-Action-Required (CAR)

CAR fires when the recon-and-scan cascade exhausts Layer 3 (target repos), Layer 2 (mounted ISO), and Layer 1 (vendored bundle) without finding a needed package. `GET /api/customer-action` lists pending targets; `POST /api/customer-action/{target}/retry` re-probes for media markers (`Packages/`, `pool/`, `suse/`, `repodata/`, `media.1/`) after the customer mounts install media.

10 · Resource groups & control overrides

Resource groups bundle hosts with shared credentials so an operator can scan/remediate "all DCs" or "all DMZ Linux" by group name. Passwords are redacted in all responses.

`POST` `/api/resource-group`

Create or update a group; `name` validated against `[A-Za-z0-9 _.-]{1,64}`.

```
{"name": "DC-Servers", "hosts": ["10.0.0.10", "10.0.0.11"],
 "username": "Administrator", "password": "<pw>",
 "os_type": "windows", "description": "Domain controllers"}
```

Control overrides

Override records flow into the SSP and POA&M as documented exceptions or alternative implementations.

```
{"control_id": "AC.L1-3.1.1", "target": "all",
 "type": "documented_exception",
 "justification": "...",  // 20..4000 chars
 "evidence_ref": "POL-AC-001", "author": "compliance@acme.com"}
```

Note — DELETE refuses while any scan is in flight (409 `scan_in_progress`) -- overrides feed scan-result interpretation.

11 · Licensing

The license.key `key_material` field is an AEAD-wrapped envelope (base64 of canonical JSON with `nonce`, `scheme`, `wrapped`). The customer-facing surface (import / activate / status) is unchanged; the binary handles the envelope unwrap internally. A license issued for one machine cannot decrypt on another.

`POST` `/api/import-license`

Import a license.key file (raw text or base64-wrapped). Distinct error codes for `expired` / `fingerprint_mismatch` / `bad_signature`. `REMEDIATION_AVAILABLE` updates at runtime -- no restart required.

`POST` `/api/claim-license`

Exchanges a one-time claim token (JWT from the post-purchase email) for a signed license bound to this machine's fingerprint. FF POSTs the token and its local fingerprint upstream to `license-api/api/licenses/claim`, receives signed license content, and runs the same import path as `/api/import-license`.

> **Note —** 503: `claim_not_configured` if `LICENSE_API_URL` is unset on this build (current prod builds default to unset).

`GET` `/api/license-status`

License tier, expiry, `days_remaining`, request code. Tier is downgraded to free when expired so the dashboard never shows a stale Pro badge. Always returns 200 -- check the `licensed` field.

`GET` `/api/evidence-status`

Whether collected evidence is still valid. Set to invalid when any rollback succeeds; cleared when a fresh scan completes.

12 · Scan scheduler & definition updates (Pro / Beta)

> **Note —** Beta -- the scheduler endpoints are wired and the module is loaded, but the scan callback is currently a stub returning `{"status": "not_implemented"}`. `/api/scheduler/start` refuses with 503 `scheduler_callback_stub` when it detects the stub. Status, config GET/POST, stop, and history are fully functional. Do not advertise scheduled scans as production-ready until the callback is wired.

`GET` `/api/scheduler/status`

Scheduler state (enabled, interval, running, next_run, last_run, history_count) plus definition status (version, age_days, stale). Available to all users including free tier.

`GET` `POST` `/api/scheduler/config`

GET returns current config; POST updates it. Fields: `enabled`, `interval` (daily/weekly/biweekly/monthly), `scan_hour`, `scan_minute`, `targets`, `credentials`, `auto_evidence`.

Definition updates

Definition packages contain updated check logic, remediation commands, and pass/fail thresholds. Packages are Ed25519-signed by DenseDefense. Connected systems auto-pull before each scan; air-gapped systems import manually via `POST /api/definitions/import`. The engine rejects unsigned, tampered, or downgrade packages.

13 · Operator settings, outreach & warranty

Persistent operator preferences (exclusion list, fleet worker cap), customer self-serve outreach, and the opt-in online warranty check. All are local -- outreach persists to `call_requests.jsonl` on the FF host; warranty calls densedefense.com only when explicitly enabled.

Excluded targets

`GET/POST/DELETE /api/settings/excluded-targets` -- operator-defined out-of-scope IPs/hostnames. Discovery (Network / AD / Jump Host) filters matches on every engagement; the count surfaces in discovery stats as `out_of_scope_filtered`. Persisted to `/var/lib/fortefide/excluded_targets.json` (atomic tmp+rename); survives a dpkg-purge unless the operator removes the file.

Customer outreach

`POST /api/request-call` is a self-serve outreach form persisting to `call_requests.jsonl`; `GET /api/request-call/log` is the operator-side read. Not auth-gated at this layer -- bind to an isolated management network for production.

Online warranty check (opt-in)

`GET /api/warranty-status`, `GET/POST /api/warranty-settings` (`{"enabled": true|false}`, toggling triggers an immediate check), and `POST /api/warranty-check` (400 `opt_in_disabled` if opt-in is off).

Scan keys

`POST /api/generate-scan-keys` generates the SSH key pair and X.509 client cert used for scan/prep/teardown auth. Refuses with 409 `prepared_endpoints_would_orphan` if any prepared endpoints exist (regenerating would silently invalidate every authorized_keys entry on those targets) -- override with `{"force_regen": true}`. `GET /api/scan-keys-status` checks whether keys/certs exist on disk.

14 · Engagement lifecycle

`GET` `/api/engagement/pregame`

Returns the catalogue-coverage report an operator must acknowledge before starting prepare/scan. Counts are re-derived from the live CMMC matrix on every call -- never a cached or hand-typed source. Optional query `?target=<ip>` (repeatable) adds per-target applicability.

```
200: {"total_nist_canonical": 110,
      "automated_total_unique_nist": 79,
      "manual_only_family": 31, "not_implemented": 0,
      "per_os_gaps": [...], "as_of": "<iso8601>",
      "matrix_source": "cmmc_matrix.CMMC_MATRIX"}
```

`POST` `/api/engagement/pregame/acknowledge`

Records an operator's acknowledgment of the pre-engagement coverage report. Appends a chain-of-custody record to `journal/pregame_acks.jsonl`; the post-mortem and evidence package later surface the acknowledged gaps. Requires `operator_name` (2-120 chars), `engagement_id`, and `justification` (≥10 chars).

`GET` `/api/engagement/{id}/status`

Single source of truth for engagement phase: aggregates the baseline scan + linked remediation jobs into one `phase` value: `baseline_running → baseline_complete → remediation_running → remediation_complete → unknown`.

`GET` `/api/engagement/{id}/postmortem`

Per-target per-control failure analysis for a completed engagement. Customer-protection rule: every one of the 110 canonical NIST 800-171r2 controls appears with a determination -- a control that was not evaluated is explicitly marked `MANUAL_REQUIRED` with a reason (`not_implemented`, `scan_skipped`, etc.), never silently omitted. Required input to evidence-package and POA&M generation.

`POST` `/api/attestation/{family}`

Uploads a signed manual-control attestation PDF for one CMMC family. Manual families (AT, PE, PS, CA, IR, MA, MP, SC controls marked organizational) emit `ATTESTATION_REQUIRED` rather than a pass/fail. Multipart fields: `file` (PDF, must start with `%PDF`), `scan_id`, `signer_name`, `signer_title`, `signed_date` (ISO date). Bundled into the signed evidence ZIP and surfaced in the sealed attestation index; does not affect the automated score.

15 · Admin API (admin_api.py)

Vendor-side license provisioning service. Runs on DenseDefense infrastructure (port 9410) behind a TLS reverse proxy in production. Bearer-token authenticated (`ADMIN_API_KEY`) except the webhook (HMAC-SHA256). **Never compiled into a customer-facing build.** `ADMIN_API_DEMO=true` bypasses auth and binds to 0.0.0.0 for the public pricing-page demo flow only.

`POST` `/api/provision`

Provision a single license. Accepts full provisioning fields or pricing-page fields with auto-fill. Sends the license.key by SMTP if `email` is supplied and `SMTP_USER` / `SMTP_PASS` are set.

`POST` `/api/webhook/payment`

Payment processor webhook. HMAC-SHA256 signature validated against `WEBHOOK_SECRET`. Headers checked in order: `X-Signature-256`, `Stripe-Signature`, `X-Webhook-Signature`, `Paddle-Signature` (Stripe's `t=...,v1=...` format is extracted automatically). Only `payment.completed` events are processed; others return 200 `status: ignored`.

`GET` `/api/license/{org_id}/download`

Download a provisioned license.key. Auth: Bearer `ADMIN_API_KEY`.

`GET` `/api/provision/log`

Query the provision audit trail (JSONL). Optional filters: `?org_id`, `?product`, `?action`, `?limit` (default 100). Most-recent-first.

ForteFide v26.07.27.1120 -- REST API reference generated from the live route table in app_source.py and admin_api.py. Automation/remediation commands themselves are stripped before serialization on every matrix/control response -- they are never exposed over HTTP.

DenseDefense · CMMC 2.0 / NIST SP 800-171 compliance automation. "Keys are Security."