ForteFide

API Reference

Scope	REST API Documentation for Integration & Automation
Version	1.4.2
Date	March 2026
Base URL	http://<host>:5000

Table of Contents

1. Overview

The ForteFide REST API provides programmatic access to all scanner, discovery, remediation, and management features. It is designed for integration with SIEM platforms, GRC tools, CI/CD pipelines, and custom automation scripts.

All responses include Cache-Control: no-cache, no-store, must-revalidate headers. CORS is enabled globally. All endpoints return JSON unless otherwise noted (PDF, CSV, and JSON file exports).

Conventions

- Base URL: http://<host>:5000
- All request bodies are JSON (Content-Type: application/json)
- All responses are JSON unless noted (PDF/CSV/JSON file downloads)
- No authentication required (designed for air-gapped deployment)
- IDs are sequential integers, starting at 1 per session
- Completed scans persist to AES-256-GCM encrypted SQLite and auto-restore on restart. In-memory state (discoveries, preparations, remediation jobs) is cleared on restart.

2. Endpoint Summary

Complete list of all 55 API routes organized by category.

System (6 endpoints)

Method	Endpoint	Description
GET	/api/capabilities	Feature flags for running instance
GET	/api/health	Service health, uptime, state counts
GET	/api/logs	Activity log (circular buffer, max 3000)
POST	/api/import-license	Import license key to activate Pro
POST	/api/activate	Activate Pro with activation code
GET	/api/license-status	License tier, status, and request code

Scanning (9 endpoints)

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 all scans
GET	/api/scan/history	Score history for trend charting
GET	/api/scan/{id}/delta/{prev}	Compare two completed scans
GET	/api/scan/{id}/pdf	Export results as PDF
GET	/api/scan/{id}/csv	Export results as CSV
GET	/api/scan/{id}/json	Export results as JSON file

Discovery (3 endpoints)

Method	Endpoint	Description
POST	/api/discover	Start network/AD/proxy discovery
GET	/api/discover/{id}	Get discovery state and results
POST	/api/discover/{id}/abort	Stop a running discovery

Control Matrix (2 endpoints)

Method	Endpoint	Description
GET	/api/matrix	Full CMMC matrix (filterable)
GET	/api/matrix/{control_id}	Single control entry

Remediation & Rollback -- Pro (5 endpoints)

Method	Endpoint	Description
POST	/api/remediate	Remediate single control
POST	/api/remediate-batch	Batch remediate multiple controls
POST	/api/rollback	Undo single remediation
POST	/api/rollback-batch	Undo multiple remediations
GET	/api/remediation-history	Session remediation history

Endpoint Preparation -- Pro (5 endpoints)

Method	Endpoint	Description
POST	/api/prepare-endpoint	Create service account on target
GET	/api/prepare-endpoint/{id}	Poll preparation status
POST	/api/teardown-endpoint	Remove service account
GET	/api/svc-account/{target}	Get stored credentials
GET	/api/svc-accounts	List all prepared endpoints

3. System Endpoints

GET `/api/capabilities`

Returns feature flags for the running instance. Use this to determine which modules are available before calling their endpoints.

Response: 200

```
{
  "scanner": true,
  "remediation": false,
  "discovery": true,
  "ldap": true,
  "cert_auth": false,
  "ssh_key": false,
  "client_cert": false,
  "version": "1.4.2"
}
```

Field	Type	Description
scanner	bool	Always true
remediation	bool	Pro module installed and licensed
discovery	bool	Network/AD discovery available
ldap	bool	AD/LDAP discovery available
cert_auth	bool	Certificate auth files exist
ssh_key	bool	SSH key at keys/scan_key
client_cert	bool	Client cert at keys/client_cert.pem
version	string	Product version string

GET /api/health

Service health check. Returns uptime, active operation counts, and in-memory state sizes. Suitable for monitoring scripts and load balancer probes.

Response: 200

```
{
  "status": "ok",
  "started": "2026-03-18T14:00:00Z",
  "uptime_seconds": 3600,
  "active_scans": 1,
  "active_discoveries": 0,
  "total_scans": 5,
  "total_discoveries": 2,
  "log_entries": 142,
  "remediation_actions": 3,
  "service_accounts": 1
}
```

Field	Type	Description
status	string	Always 'ok' if service is running
started	string	UTC ISO 8601 server start time
uptime_seconds	int	Seconds since server start
active_scans	int	Scans currently queued or running
active_discoveries	int	Discoveries currently running
total_scans	int	Total scans in session (all states)
total_discoveries	int	Total discoveries in session
log_entries	int	Activity log size (max 3000)
remediation_actions	int	Remediations applied this session
service_accounts	int	Prepared service accounts in memory

GET /api/logs

Returns activity log entries from a circular buffer (max 3000 entries). Use the 'since' parameter for efficient polling.

Query Parameters:

Param	Type	Default	Description
since	int	0	Return entries with ID > this value

Response: 200

```
[{
  "id": 42,
  "ts": "14:23:01",
  "cat": "SCAN",
  "msg": "Scanning 192.168.30.10 (WinRM): 55/110 controls (50%)",
  "status": "info"
}]
```

Log Categories:

Category	Description
SYSTEM	Server startup, module loading
SCAN	Scan start/complete/error
PROGRESS	Scan progress updates
REMEDIATE	Remediation actions
ROLLBACK	Rollback actions
DISCOVERY	Discovery operations
PREPARE	Endpoint preparation

POST `/api/import-license`

Import a license.key file to activate Pro features without restarting the service. The content field accepts base64-encoded or raw license text. On success, the license is written to ProgramData/ForteFide/license.key and the REMEDIATION_AVAILABLE flag is updated at runtime.

Request Body:

```
{
  "content": "<base64 or raw license text>"
}
```

Response: 200

```
{
  "success": true,
  "license": {
    "org_id": "ACME-001",
    "tier": "pro",
    "max_endpoints": 100,
    "issued": "2026-03-20",
    "expires": "2027-03-20"
  }
}
```

Response Fields:

Field	Type	Description
success	bool	true if license was imported and validated
license.org_id	string	Organization identifier from the license
license.tier	string	License tier (pro)
license.max_endpoints	int	Maximum endpoints allowed by license
license.issued	string	License issue date
license.expires	string	License expiration date

Notes:

- License is persisted to ProgramData/ForteFide/license.key
- REMEDIATION_AVAILABLE is updated at runtime -- no restart required
- Replaces any previously installed license

Errors:

- 400: Invalid or missing content field (invalid format)
- 400: License signature verification failed (validation failed)

POST `/api/activate`

Activate Pro remediation with a one-time activation code. The code is AES-256-GCM encrypted and cryptographically bound to the build key and the license ID. A valid license must be installed first via /api/import-license.

Request Body:

```
{
  "activation_code": "FIDE-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX"
}
```

Response: 200

```
{"success": true, "message": "Activation successful"}
```

Notes:

- Activation code is AES-256-GCM encrypted, bound to build key + license ID
- Each code is single-use and cannot be reused after successful activation
- Requires a valid license to be installed first

Errors:

- 400: Invalid or expired activation code (invalid code)
- 500: Remediation module not available (module not available)

GET `/api/license-status`

Get the current license status, tier, and machine request code. The request code is a device fingerprint used for generating machine-bound licenses. Call this endpoint to determine whether Pro features are available and to obtain the request code needed when requesting a license from DenseDefense.

Response: 200

```
{
  "licensed": true,
  "tier": "pro",
  "info": {
    "org_id": "ACME-001",
    "max_endpoints": 100,
    "issued": "2026-03-20",
    "expires": "2027-03-20"
  },
  "request_code": "FIDE-A1B2-C3D4-E5F6"
}
```

Response (unlicensed):

```
{
  "licensed": false,
  "tier": "free",
  "info": null,
  "request_code": "FIDE-A1B2-C3D4-E5F6"
}
```

Response Fields:

Field	Type	Description
licensed	bool	true if a valid license is installed
tier	string	Current tier: 'free' or 'pro'
info	object|null	License details (null if unlicensed)
info.org_id	string	Organization identifier
info.max_endpoints	int	Maximum licensed endpoints
info.issued	string	License issue date
info.expires	string	License expiration date
request_code	string	Device fingerprint for machine-bound licensing

Notes:

- The request_code is the device fingerprint for machine-bound license generation
- Always returns 200 -- use the 'licensed' field to check activation state
- The request_code is stable across restarts for the same machine

4. Scanning Endpoints

POST /api/scan

Start a compliance scan against one or more endpoints. Supports two target formats: a comma-separated string or an array of target objects.

Request Body (string format):

```
{
  "targets": "192.168.30.10,192.168.30.11",
  "mode": "full",
  "username": "fortefide-svc",
  "password": "P@ssw0rd",
  "os_type": "windows",
  "auth_mode": "password"
}
```

Request Body (array format):

```
{
  "targets": [
    {
      "host": "192.168.30.10",
      "os_type": "windows",
      "username": "fortefide-svc",
      "password": "P@ssw0rd",
      "auth_mode": "password"
    }
  ],
  "mode": "full"
}
```

Parameters:

Field	Type	Required	Default	Description
targets	string|array	Yes	-	Comma IPs or target objects
mode	string	No	quick	quick (ports) or full (110 controls)
username	string	For full	-	WinRM/SSH username
password	string	For full	-	WinRM/SSH password
os_type	string	No	windows	windows or linux
auth_mode	string	No	password	password or certificate

Response: 200

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

Errors:

- 400: No targets specified
- 400: Certificate files not found (auth_mode=certificate)

GET /api/scan/{scan_id}

Get the full scan state including all host results, control assessments, severity counts, and compliance score. Poll this endpoint to track scan progress.

Response: 200

```
{
  "id": 1, "status": "complete", "progress": "Complete",
  "started": "2026-03-18T14:23:00",
  "completed": "2026-03-18T14:25:30",
  "targets": ["192.168.30.10"],
  "mode": "full", "os_type": "windows", "auth_mode": "password",
  "compliance_score": 72.5,
  "total_controls": 110, "passed_controls": 80,
  "severity_counts": {"critical": 2, "high": 8, "medium": 12, "low": 5},
  "failed_controls": ["AC.1.001", "SC.1.175"],
  "family_breakdown": {"AC": 5, "SC": 3},
  "hosts": [{
    "ip": "192.168.30.10", "hostname": "DC01",
    "os_guess": "Windows Server",
```

```
    "open_ports": [{"port": 5985, "service": "WinRM"}],
    "control_results": [{
      "control_id": "AC.1.001", "status": "compliant",
      "title": "Limit information system access...",
      "severity": "high", "family": "AC",
      "remediation_steps": ["Step 1", "Step 2"],
      "automation_safe": true, "rollback_available": true,
      "risk_level": "medium", "impact_warning": "May affect user access"
    }]
  }]
}
```

Scan Status Values:

Status	Description
queued	Scan accepted, waiting to start
running	Scan in progress
complete	All targets scanned, results available

Errors: 404: Scan not found

POST /api/scan/{scan_id}/abort

Signal a running scan to stop. The scan completes with partial results for any hosts already scanned.

Response: 200

```
{"ok": true, "scan_id": 1}
```

Errors:

- 404: Scan not found
- 400: Scan already finished

GET /api/scans

List all scans in the current session. Returns an array of full scan objects (same structure as GET /api/scan/{id}).

Response: 200 -- Array of scan objects

GET /api/scan/history

Scan summaries optimized for trend graphing. Only includes completed authenticated scans. Returns per-family pass rates for breakdown charts.

Response: 200

```
[{
  "scan_id": 1,
  "date": "2026-03-18T14:25:30",
  "targets": ["192.168.30.10"],
  "score": 72.5, "passed": 80, "failed": 30, "total": 110,
  "severity_counts": {"critical": 2, "high": 8},
  "family_scores": {"AC": {"passed": 12, "total": 22}}
}]
```

GET /api/scan/{scan_id}/delta/{prev_id}

Compare two completed scans to identify improvements and regressions. Returns lists of newly passing, newly failing, still failing, and still passing controls.

Response: 200

```
{
  "current_scan": 2, "previous_scan": 1,
  "current_score": 78.2, "previous_score": 72.5,
  "score_change": 5.7,
  "newly_passing": [
    {"control": "AC.1.001", "description": "...",
     "family": "AC", "severity": "high"}
  ],
  "newly_failing": [], "still_failing": [], "still_passing": [],
  "current_date": "2026-03-18T15:00:00",
  "previous_date": "2026-03-18T14:25:30"
```

```
}
```

Errors:

- 404: One or both scans not found
- 400: One or both scans not complete

Export Endpoints

Three export formats are available for completed scans. All return downloadable files with Content-Disposition headers.

GET /api/scan/{scan_id}/pdf

Export scan results as a formatted PDF report with cover page, executive summary, severity breakdown, and per-host detailed findings.

- Content-Type: application/pdf
- Filename: ForteFide_Report_{scan_id}.pdf

Errors: 404: Scan not found or not complete

GET /api/scan/{scan_id}/csv

Export scan results as CSV for import into spreadsheets, SIEMs, or GRC platforms.

Query Parameters:

Param	Type	Default	Description
scope	string	findings	findings (failed only) or all (every control)

CSV Columns:

Host, Control ID, Family, Severity, Status, Title, Detail, Risk Level, Remediation Steps

- Content-Type: text/csv
- Filename: ForteFide_Report_{scan_id}.csv

Errors: 404: Scan not found or not complete

GET /api/scan/{scan_id}/json

Export the full scan object as a downloadable JSON file. Same data as GET /api/scan/{id} but with Content-Disposition for file download.

- Content-Type: application/json
- Filename: ForteFide_Report_{scan_id}.json

Errors: 404: Scan not found or not complete

5. Discovery Endpoints

POST /api/discover

Start host discovery using one of three methods: network scan, Active Directory LDAP query, or proxy/jump host discovery.

Network Discovery:

```
{"method": "network", "cidr": "192.168.30.0/24"}
```

Active Directory Discovery:

```
{
  "method": "ad",
  "dc_ip": "192.168.30.10",
  "domain": "lab.local",
  "dc_username": "Administrator",
  "dc_password": "P@ssw0rd"
}
```

Proxy/Jump Host Discovery:

```
{
  "method": "proxy",
  "proxy_host": "192.168.30.160",
  "proxy_port": 22,
  "proxy_username": "violet",
```

```
  "proxy_password": "password",
  "proxy_target": "network",
  "cidr": "10.0.0.0/24"
}
```

Parameters by Method:

Field	network	ad	proxy
method	Required	Required	Required
cidr	Required	-	If proxy_target=network
dc_ip	-	Required	If proxy_target=ad
domain	-	Required	If proxy_target=ad
dc_username	-	Required	If proxy_target=ad
dc_password	-	Required	If proxy_target=ad
proxy_host	-	-	Required
proxy_port	-	-	Required (default: 22)
proxy_username	-	-	Required
proxy_password	-	-	Required
proxy_target	-	-	network or ad

Response: 200

```
{"discovery_id": 1, "status": "running", "method": "network"}
```

GET `/api/discover/{disc_id}`

Get discovery state and results. Poll this endpoint until status is 'complete'.

Response: 200

```
{
  "id": 1, "method": "network",
  "status": "complete", "progress": "Complete",
  "hosts": [{
    "ip": "192.168.30.10", "hostname": "DC01",
    "os": "Windows Server 2022", "source": "network",
    "ports": [53, 88, 389, 445, 5985]
  }],
  "stats": {
    "total_hosts": 5, "windows_hosts": 3,
    "linux_hosts": 1, "unknown_hosts": 1, "enabled": 5
  },
  "error": null
}
```

POST `/api/discover/{disc_id}/abort`

Stop a running discovery. Returns partial results.

Response: 200

```
{"ok": true, "discovery_id": 1}
```

6. CMMC Control Matrix

GET `/api/matrix`

Returns the full CMMC/NIST 800-171 control matrix. Automation commands are stripped for security -- remediation commands are never exposed over HTTP. Supports server-side filtering by family, severity, and text search.

Query Parameters (all optional, combinable):

Param	Type	Description
family	string	Filter by family code (e.g. AC, SC, AU)
severity	string	Filter by level (critical, high, medium, low, info)
q	string	Case-insensitive search across title + description

Example Queries:

- /api/matrix?family=AC -- all 22 Access Control controls
- /api/matrix?severity=critical -- critical controls only
- /api/matrix?family=SC&severity=high -- high-severity SC controls
- /api/matrix?q=encryption -- controls mentioning encryption

Response: 200 -- Object keyed by control ID

```
{
  "AC.1.001": {
    "title": "Limit information system access to authorized users",
    "family": "AC", "severity": "high",
    "description": "...",
    "remediation_steps": ["Step 1", "Step 2"],
    "remediation_impact": "...",
    "expected_outcome": "...",
    "risk_if_unaddressed": "...",
    "check_description": "...",
    "nist_ref": "3.1.1",
    "references": []
  }
}
```

Control Fields:

Field	Type	Description
title	string	Control title
family	string	Family code (AC, SC, AU, etc.)
severity	string	critical, high, medium, low, info
description	string	Full control description
remediation_steps	array	Step-by-step remediation guidance
remediation_impact	string	Impact of applying remediation
expected_outcome	string	Expected state after remediation
risk_if_unaddressed	string	Risk of leaving non-compliant
check_description	string	How the scanner checks this control
nist_ref	string	NIST 800-171 reference number
references	array	External reference URLs

GET `/api/matrix/{control_id}`

Returns a single control entry. Same fields as the matrix response above.

Errors: 404: Control not found

7. Remediation Endpoints (Pro)

All remediation endpoints require the Pro module with a valid license. Unlicensed requests return 403: {"error": "Remediation module not installed", "upgrade": true}.

POST `/api/remediate`

Execute automated remediation for a single control on a target endpoint. The command is decrypted in memory, executed via WinRM (Windows) or SSH (Linux), and the result is recorded in remediation history for rollback tracking.

Request Body:

```
{
  "control_id": "AC.1.001",
  "target": "192.168.30.10",
  "username": "fortefide-svc",
  "password": "P@ssw0rd",
  "os_type": "windows"
}
```

Response: 200

```
{
  "success": true,
  "control_id": "AC.1.001",
  "target": "192.168.30.10",
  "stdout": "Registry key set successfully",
  "stderr": "",
  "expected_outcome": "Access control policy enforced",
  "rollback_available": true
}
```

POST `/api/remediate-batch`

Execute remediation for multiple controls on a single target. Controls are processed sequentially. Each result is recorded individually in remediation history.

Request Body:

```
{
  "target": "192.168.30.10",
  "username": "fortefide-svc",
  "password": "P@ssw0rd",
  "os_type": "windows",
  "control_ids": ["AC.1.001", "AC.1.002", "SC.1.175"]
}
```

Response: 200

```
{
  "succeeded": 2, "failed": 1,
  "results": [{
    "control_id": "AC.1.001", "success": true,
    "stdout": "...", "stderr": "",
    "expected_outcome": "...",
    "rollback_available": true
  }]
}
```

8. Rollback Endpoints (Pro)

Rollback reverses previously applied remediations. Rollback is only available for controls remediated in the current session (in-memory history). 103 of 110 controls have rollback commands (62 Windows + 41 Linux); the remaining 7 are non-reversible.

POST `/api/rollback`

Undo a previously applied remediation for a single control.

Request Body:

```
{
  "control_id": "AC.1.001",
  "target": "192.168.30.10",
  "username": "fortefide-svc",
  "password": "P@ssw0rd",
  "os_type": "windows"
}
```

Response: 200

```
{
  "success": true,
  "control_id": "AC.1.001",
  "target": "192.168.30.10",
  "stdout": "Registry key restored",
  "stderr": ""
}
```

Errors: 400: Control was not remediated on this target in the current session

POST /api/rollback-batch

Undo multiple remediations on a single target.

Request Body:

```
{
  "target": "192.168.30.10",
  "username": "fortefide-svc",
  "password": "P@ssw0rd",
  "os_type": "windows",
  "control_ids": ["AC.1.001", "AC.1.002"]
}
```

Response: 200

```
{
  "succeeded": 2, "failed": 0,
  "results": [
    {"control_id": "AC.1.001", "success": true, "stderr": ""}
  ]
}
```

GET /api/remediation-history

Returns the current session's remediation history. Each entry tracks which controls were remediated on which targets, enabling targeted rollback.

Response: 200

```
{
  "192.168.30.10:AC.1.001": {
    "timestamp": "2026-03-18T14:30:00",
    "os_type": "windows",
    "rollback_available": true
  }
}
```

9. Endpoint Preparation (Pro)

Endpoint preparation creates a dedicated service account on target systems for scanning and remediation. Credentials are stored in memory for the session. Teardown removes the account when no longer needed.

Transport Options (v1.4.2)

Both prepare and teardown accept an optional 'transport' field to control how ForteFide connects to Windows targets:

Value	Port	Description
auto	22/5985	Default. Probes ports, prefers SSH > PSRP > WinRM
winrm	5985	WinRM with NTLM auth (pywinrm). ~8 KB encoded command limit
ssh	22	OpenSSH Server with -EncodedCommand (paramiko). No length limit
psrp	5985	PowerShell Remoting Protocol (pypsrp). No length limit

The response includes a 'transport' field showing which method was actually used. Linux targets always use SSH.

POST /api/prepare-endpoint

Create a service account on a target endpoint. Runs asynchronously -- poll the returned prep_id for status.

Request Body:

```
{
  "target": "10.0.1.50",
  "username": "Administrator",
  "password": "P@ssw0rd",
  "os_type": "windows",
  "svc_username": "fortefide-svc",
  "transport": "auto"
}
```

Response: 200

```
{"prep_id": 1, "status": "running", "target": "10.0.1.50"}
```

GET /api/prepare-endpoint/{prep_id}

Poll preparation status. Password is redacted in responses.

Response: 200

```
{
  "id": 1, "target": "192.168.30.10",
  "os_type": "windows",
  "status": "complete", "progress": "Done",
  "result": {"success": true, "svc_username": "fortefide-svc"}
}
```

POST /api/teardown-endpoint

Remove a previously created service account from a target. Supports password-based and inventory-based (key/cert) authentication.

Request Body (password auth):

```
{
  "target": "10.0.1.50",
  "username": "Administrator",
  "password": "P@ssw0rd",
  "os_type": "windows",
  "transport": "auto"
}
```

Request Body (inventory/key-based auth):

```
{
  "target": "10.0.1.50",
  "os_type": "windows",
  "auth_mode": "inventory",
  "transport": "ssh"
}
```

Response: 200

```
{"success": true, "result": "Service account removed", "error": null}
```

GET /api/svc-account/{target}

Get stored service account credentials for a specific target. Returns the full password (use only in secure contexts).

Response: 200

```
{
  "exists": true,
  "username": "fortefide-svc",
  "password": "generated-password",
  "os_type": "windows"
}
```

Errors: 404: No service account for this target

GET /api/svc-accounts

List all prepared endpoints. Passwords are redacted in this listing.

Response: 200

```
[{"target": "192.168.30.10", "username": "fortefide-svc", "os_type": "windows"}]
```

10. Scan Scheduler (Pro)

The scan scheduler runs compliance scans automatically on a configurable cadence. Before each scan, it checks for definition updates (auto-pull on connected systems, reminder on air-gapped). Requires a valid license -- returns 403 without one.

GET /api/scheduler/status

Returns scheduler state (enabled, interval, next/last run) and definition status. Available to all users including free tier.

Response: 200

```
{
  "scheduler": {"enabled": true, "interval": "weekly",
    "running": true, "next_run": "2026-03-30T02:00:00",
    "last_run": "2026-03-23T02:00:01Z", "history_count": 4},
  "definitions": {"version": "DEF-2026.03.25-001",
    "age_days": 2.1, "stale": false}
}
```

GET/POST /api/scheduler/config

GET returns current config. POST updates config (requires license). Fields: enabled (bool), interval (daily/weekly/biweekly/monthly), scan_hour (0-23), scan_minute (0-59), targets (array), credentials (object), auto_evidence (bool).

POST Body:

```
{"enabled": true, "interval": "weekly", "scan_hour": 2,
 "targets": ["192.168.30.10", "192.168.30.11"],
 "credentials": {"username": "Administrator",
   "password": "...", "domain": "lab.local", "os_type": "windows"}}
```

POST /api/scheduler/start

Start the scheduler background thread. Requires license.

POST /api/scheduler/stop

Stop the scheduler background thread. Requires license.

GET /api/scheduler/history

Returns scan history from scheduled runs. Each entry: timestamp, status, score, passed/total, scan_id. Retained for 90 days.

11. Definition Updates (Pro)

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.

GET /api/definitions/status

Returns current definition version, age, staleness, and reminder state. Available to all users.

Response: 200

```
{"version": "DEF-2026.03.25-001", "installed_at": "2026-03-25T14:37:40Z",
 "age_days": 2.1, "stale": false, "critical": false,
 "auto_update": false, "reminder_active": false}
```

POST /api/definitions/import

Import a signed definition package. Accepts JSON body or multipart file upload. Requires license. Rejects unsigned, tampered, or downgrade packages.

JSON Body:

```
{"version": "DEF-2026.03.25-001", "timestamp": "...",
 "check_updates": [...], "remediation_updates": [...],
 "changelog": "Modified AC.1.003 firewall check...",
 "signature": "<base64 Ed25519 signature>"}
```

GET `/api/definitions/history`

Returns definition update history: old/new version, timestamp, who imported, changelog.

POST `/api/definitions/dismiss-reminder`

Dismiss the stale-definition reminder for 24 hours (or custom). Body: {"hours": 24}

12. State Model

ForteFide v1.4.2 uses session persistence with AES-256-GCM encrypted SQLite. Completed scan results are saved to the encrypted database and auto-restored on restart. In-memory state (discoveries, preparations, remediation jobs) is still cleared on restart. This hybrid approach preserves compliance history while ensuring that transient credentials and session data are never persisted to disk.

State	Lifetime	Max Size	Cleared On
Scans (completed)	Persistent	Unlimited	Encrypted SQLite, survives restart
Scans (in-progress)	Session	Unlimited	Service restart
Discoveries	Session	Unlimited	Service restart
Preparations	Session	Unlimited	Service restart
Service accounts	Session	Unlimited	Service restart
Remediation history	Session	Unlimited	Service restart
Activity log	Session	3000 entries	Circular buffer

Completed scan results are automatically persisted to AES-256-GCM encrypted SQLite and restored on startup. To share results externally, use the PDF, CSV, or JSON export endpoints.

13. Security Notes

#	Control	Description
1	No authentication	Designed for air-gapped/internal deployment
2	Commands stripped	/api/matrix never exposes remediation commands
3	Cache disabled	All responses include no-cache headers
4	Password redaction	prepare-endpoint/{id} and svc-accounts redact
5	CORS enabled	Accepts requests from any origin
6	Encrypted persistence	Scan results in AES-256-GCM SQLite; credentials never written to disk

Recommendation: Deploy ForteFide on an isolated management network segment. The API has no authentication and should not be exposed to untrusted networks.

14. Integration Examples

Scan and Export (curl)

```
# Start a full scan
curl -X POST http://localhost:5000/api/scan \
  -H "Content-Type: application/json" \
  -d '{"targets":"192.168.30.10","mode":"full",\
       "username":"fortefide-svc","password":"P@ss",\
       "os_type":"windows"}'

# Poll until complete
curl http://localhost:5000/api/scan/1

# Export results
curl -O http://localhost:5000/api/scan/1/pdf
curl -O http://localhost:5000/api/scan/1/csv
curl -O http://localhost:5000/api/scan/1/json
```

Python Integration

```
import requests, time

BASE = "http://fortefide-host:5000"

# Check service health
health = requests.get(f"{BASE}/api/health").json()
print(f"Uptime: {health['uptime_seconds']}s")

# Start scan
r = requests.post(f"{BASE}/api/scan", json={
    "targets": "192.168.30.10",
    "mode": "full",
    "username": "fortefide-svc",
    "password": "P@ssw0rd",
    "os_type": "windows"
})
scan_id = r.json()["scan_id"]

# Poll until complete
while True:
    scan = requests.get(f"{BASE}/api/scan/{scan_id}").json()
    if scan["status"] == "complete":
        break
    time.sleep(5)

print(f"Score: {scan['compliance_score']}%")

# Download CSV
csv_data = requests.get(f"{BASE}/api/scan/{scan_id}/csv")
with open("report.csv", "wb") as f:
    f.write(csv_data.content)
```

SIEM Integration (CSV ingest)

```
# Export findings CSV for SIEM import
curl "http://localhost:5000/api/scan/1/csv?scope=findings" \
  -o findings.csv

# Export all controls (passed + failed) for compliance audit
curl "http://localhost:5000/api/scan/1/csv?scope=all" \
  -o full_audit.csv
```

Discovery + Scan Workflow

```
# Discover hosts on the network
curl -X POST http://localhost:5000/api/discover \
  -H "Content-Type: application/json" \
  -d '{"method":"network","cidr":"192.168.30.0/24"}'

# Poll discovery
curl http://localhost:5000/api/discover/1

# Use discovered hosts as scan targets
curl -X POST http://localhost:5000/api/scan \
  -H "Content-Type: application/json" \
  -d '{"targets":"192.168.30.10,192.168.30.11",\
       "mode":"full","username":"fortefide-svc",\
       "password":"P@ss","os_type":"windows"}'
```

Control Matrix Lookup

```
# Get all critical Access Control findings
curl "http://localhost:5000/api/matrix?family=AC&severity=critical"

# Search for encryption-related controls
curl "http://localhost:5000/api/matrix?q=encryption"

# Get details for a specific control
curl http://localhost:5000/api/matrix/AC.1.001
```