C3PAO ASSESSMENT GUIDE V26.08.13.0635 DENSEDEFENSE | FORTEFIDE

ForteFide C3PAO Assessor Guide

A reference for Certified Third-Party Assessment Organization personnel evaluating a customer environment that used ForteFide to prepare for a CMMC Level 2 assessment -- what ForteFide asserts, what a signed evidence package contains, and how to verify every artifact in it cryptographically and offline.

- Signed evidence package -- Pro license required
- Verification -- offline, no DenseDefense server access
- Manual-only controls -- assessor / customer attestation

How to use this guide

START HERE

Section 1 (Audience & Purpose) then Section 4 (Evidence Package Contents) -- what you will actually receive

VERIFY FIRST

Section 5 (Signature Verification) -- run the bundled verifier before trusting anything else in the package

WORKFLOW

Section 9 (Sample Assessor Workflow) -- a phase-by-phase intake checklist you can adapt to your C3PAO's process

CONTENTS

1 | Audience & Purpose

Audience

This guide is written for C3PAO assessors -- Certified Third-Party Assessment Organization personnel performing CMMC Level 2 certification assessments under 32 CFR Part 170. Specifically: Certified CMMC Professionals (CCP), Certified CMMC Assessors (CCA), and Lead Assessors who will encounter ForteFide-generated evidence during a customer's assessment.

Purpose

ForteFide is a customer-operated tool. It runs inside the OSC's (Organization Seeking Certification) environment, assesses systems against all **110** NIST SP 800-171 Rev 2 security requirements -- **79** via technical scan, the remaining **31** via organizational evidence and attestation -- and, optionally, remediates findings and produces a signed evidence package. As an assessor reviewing that package, you need to know:

- What ForteFide does and does not assert.
- What artifacts are inside a signed evidence package.
- How to verify the package was produced by genuine ForteFide software and has not been tampered with.
- Which trust assumptions are baked into the tool.
- Which controls ForteFide deliberately leaves to manual verification, and why.
- How to interpret the metadata in a way that supports your independent assessment determination.

What this guide is not

This guide is not assessor training. It does not interpret 32 CFR Part 170, NIST SP 800-171A assessment objectives, or the Cyber AB Assessment Process. It does not tell you whether to accept ForteFide evidence -- that judgment belongs to you and your C3PAO. It documents what ForteFide produces and how to verify it.

> **Tip -- Vendor independence.** ForteFide is a third-party tool. The OSC operates ForteFide themselves; DenseDefense does not have access to customer scan data, evidence, or signing material. The signed evidence package is self-contained: every claim it makes is verifiable locally with the bundled public key and verifier script, with no callback to DenseDefense.

2 | ForteFide Overview

What ForteFide is

ForteFide is a CMMC Level 2 / NIST SP 800-171 Rev 2 compliance preparation platform with three coupled functions:

- **Scanner** -- technically scans **79 of 110** controls across 14 NIST 800-171 r2 families against Windows and Linux targets. Read-only operation; no configuration changes during a scan.
- **Remediator** -- optional, opt-in, per-control. Applies fixes via authenticated remote sessions and captures rollback bundles before each change. Customer reviews and approves before remediation runs. The remediation catalogue auto-fixes **67** controls on Linux and **66** on Windows (**82** distinct controls across the union of both).
- **Evidence collector** -- generates the signed evidence package documented in Section 4. Signs with Ed25519 over a canonical JSON manifest and bundles a standalone verifier script.

What ForteFide is not

- **Not an assessment tool** -- ForteFide does not certify compliance, score per the DoD Assessment Methodology in a binding way, or substitute for C3PAO judgment.
- **Not continuous monitoring** -- a signed package is a point-in-time attestation. The customer's environment can change after sealing without invalidating the signature.
- **Not an EDR/SIEM** -- ForteFide does not run agents on targets, ingest logs continuously, or alert on anomalies.
- **Not a network intrusion tool** -- scans require explicit credentials and explicit target selection. There is no discovery-by-exploitation.

Architecture in brief

- **Operator workstation** -- where ForteFide is installed and run. Holds the local dashboard API, scan database, and signing material. Identified in the evidence manifest as `machine.hostname` / `machine.fingerprint`.
- **Targets** -- Windows and Linux systems in scope of the customer's CUI boundary. Targets receive a dedicated SSH service account during Prepare and have it removed at Teardown.
- **Auth channel** -- SSH-only on both platforms. Windows targets run the Microsoft OpenSSH server; PowerShell over SSH using `-EncodedCommand`. Linux uses standard OpenSSH. WinRM is supported as a fallback path on legacy Windows targets, but SSH is preferred.

- **Remediation engine** -- an optional module loaded only when a valid license is imported. The free tier scanner is fully functional but cannot remediate or produce signed evidence packages.
- **License flow** -- license import is local. ForteFide does not phone home for license validation per scan; the license blob is verified locally. Air-gapped operation is supported and is the documented deployment for high-classification environments.

Standards covered

STANDARD	COVERAGE IN FORTEFIDE
NIST SP 800-171 Rev 2	All 110 security requirements across 14 families (79 have a technical scan, 82 controls are auto-remediable across the union of Linux + Windows). Each requirement maps to one or more automated probes and/or manual-only flags.
NIST SP 800-171A	Assessment objectives drive ForteFide's evidence selection. Per-control evidence captures the command, raw output, and determination.
32 CFR Part 170	ForteFide produces an SPRS scorecard, SSP scaffold, and POA&M documents in formats compatible with Final Rule submission expectations. The C3PAO confirms suitability.
DoD Assessment Methodology v2.13	SPRS scoring weights (Critical -5, High -3, Medium -3, Low -1) applied; starting score 110.

3 | The 6-Step Customer Journey

ForteFide structures the customer's work into six steps. Each step produces specific, named artifacts. The signed evidence package (Section 4) bundles the artifacts produced from Step 3 onward. Steps 1 and 2 are setup; their artifacts are operationally useful but not part of compliance evidence.

#	STEP	PURPOSE	EVIDENCE PRODUCED
1	Recon	Discover and select targets in CUI scope.	Target list (operational, not in the evidence package).
2	Prepare	Provision a per-host service account; deploy SSH key/cert auth.	Per-target service-account manifest (operational; removed by Teardown).
3	Scan	Run the 79 scanned controls against each target.	Raw scan output, control determinations, host roll-up -- captured in `scan_data.json`.
4	Remediate (optional)	Apply fixes; capture rollback bundles before each change.	Remediation log, per-control rollback bundles, journal entries (`remediation.jsonl`).
5	Evidence	Generate the signed evidence ZIP for the assessor.	Signed evidence package -- this is what the C3PAO receives. See Section 4.
6	Out (Teardown)	Remove service accounts; close out journal; restore original machine state where possible.	Teardown log entry, journal closure record.

Signed vs. unsigned evidence

ForteFide produces two distinct evidence tiers. Only one is what the assessor expects to receive.

TIER	WHEN PRODUCED	MANIFEST	USE
Signed (C3PAO-grade)	Step 5 (Evidence) of the customer journey, from a scan run after Prepare + optional Remediate.	`signed=True`, Ed25519 signature over the canonical JSON manifest with SHA-256 per artifact.	Ship to the assessor. The package the customer paid for.
Unsigned (verification only)	Automatically, when rollback-all succeeds on any target -- ForteFide rescans those targets so the operator can verify the rollback restored the baseline.	`signed=False`, `context="post_rollback_rescan"`. No Ed25519 signature.	Operator confidence only. Reject if it appears in your evidence intake.

High-impact -- Before accepting any ForteFide evidence package, open `manifest.json` at the package root and confirm `signed=True`. If you see `signed=False` or `context="post_rollback_rescan"`, this is verification data the customer should not have shipped. Reject and request the signed package.

Why the journey shape matters to the assessor

The evidence package you receive corresponds to a specific moment in the customer's journey -- specifically Step 5, after Step 3 (or Step 3 + Step 4) has run and before Step 6 has run. The `generated_utc` timestamp on the manifest pins this moment.

- If the customer ran remediation between scan and seal, the package reflects POST-remediation state -- the remediation log and DD-Trust manifest show what changed and when.
- If the customer ran Teardown between scan and your review, no further evidence can be regenerated for that scan ID. The customer would need to re-Prepare, re-Scan, and re-Seal to address any new question. This is by design (leave-no-trace) but means assessor questions are best asked before Teardown.
- Steps 1-2 happen once per assessment campaign; Steps 3-5 may iterate (scan, find issues, remediate, rescan, reseal). A customer engagement may produce multiple signed packages with successive scan IDs; the latest is typically the one of record, but earlier packages are valid evidence of progress over time.

4 | Evidence Package Contents

A signed ForteFide evidence package is a single ZIP archive named `ForteFide_Signed_Evidence_<scan_id>.zip`.

ZIP contents (canonical)

FILENAME	FORMAT	WHAT IT IS
evidence_manifest.json	JSON	Authoritative chain-of-custody record. Lists every artifact with its SHA-256 hash. Carries the Ed25519 signature, the embedded public key, machine fingerprint, license identity, and product version.
verify_evidence.py	Python	Standalone verifier. Recomputes per-artifact SHA-256 and verifies the manifest signature. No network calls; no DenseDefense dependency. Source ships in plaintext.
ForteFide_Scan_Report_<id>.pdf	PDF	Per-scan report: targets, per-control results, scoring, raw command output excerpts.
01_System_Security_Plan.pdf	PDF	SSP scaffold. System identification, environment, family-by-family compliance summary, per-control implementations. Includes placeholders the customer must complete.
02_POA_M.pdf	PDF	Plan of Action & Milestones for NOT MET controls. Severity, target remediation date by severity, finding text.
03_Asset_Inventory.pdf	PDF	Hosts in scope; OS; per-host MET/NOT MET counts.
04_SPRS_Scorecard.pdf	PDF	SPRS score (starts at 110, weighted deductions per failed control). Family-level breakdown.
05_Control_Evidence.pdf	PDF	Per-control raw evidence: the command run, the output captured, the determination. This is the artifact-level audit trail.
08_Incident_Response_Plan.pdf	PDF	IRP scaffold. Customer completes the team roster and contact fields.
09_Training_Records.pdf	PDF	AT family attestation scaffold.
10_Attestation_Index.pdf	PDF	Per-family status of the manual (organizational) controls: which families have a customer-signed attestation and which are still ATTESTATION REQUIRED. Sealed under the same Ed25519 manifest as every other artifact.

FILENAME	FORMAT	WHAT IT IS
<FAMILY>_Attestation_Signed.pdf	PDF	Customer-signed per-family attestation PDFs for the manual-only controls. Present only for families the customer has signed. The customer attests; ForteFide does not verify the underlying control.
scan_data.json	JSON	Full raw scan record. The PDFs are renderings; this is the underlying data the customer cannot alter without invalidating the manifest hash.
remediation_log.json	JSON	Activity-log slice covering remediation events for this scan. Present only when remediation ran.
control_overrides.json	JSON	Customer-asserted overrides (compensating control, risk accepted, not applicable), each with a rationale string. Present only when overrides exist.
dd-trust-manifest.json	JSON	DD-Trust event log. Each remediation gets PRE-state, POST-state, and verdict entries; an evidence-seal event is appended at sign time.
state_change_pairs.json	JSON	Per-control verified-remediation proof. One entry per remediated control: {control_id, before, after, verified}, where verified comes from re-running the same 800-171A check that originally failed -- not from the fix's exit code. Present only when remediation ran.

What is NOT in the package

- Customer admin credentials -- used once at Prepare and discarded; never persisted.
- ForteFide signing private key -- stays on the operator workstation.
- Network captures, target log files, full disk images -- ForteFide does not collect these.
- Cross-package linkage -- each scan's package stands alone; there is no cross-scan chain in the manifest.

Evidence-seal event

Immediately before the manifest is signed, ForteFide writes a `dd-trust-manifest.json` snapshot then appends an evidence-seal event recording `scan_id` and `artifact_count`. This event is the last record in the DD-Trust log inside the ZIP. Its presence demonstrates the ZIP was

produced through the documented seal path; its absence indicates a hand-assembled ZIP, which the Ed25519 signature would also catch.

5 | Signature Verification

Signature verification is the C3PAO's primary defense against a tampered or forged evidence package. The package is self-verifying: every check runs locally and uses only the public material bundled in the ZIP.

Doctrine: License = Key = Proof

ForteFide operates under an explicit trust-anchor doctrine the C3PAO should understand. The customer's `license.key` file IS the cryptographic trust anchor. ForteFide derives the signing keys from the license payload; it does not store the derived keys separately.

- **License** = the key file in `DATA_DIR`. Possession of the file is possession of the trust anchor.
- **Key** = the HKDF-derived signing material used to sign each evidence package. Re-derivable from the license at any time.
- **Proof** = the signed evidence package that DenseDefense stands behind as truthful, provided the underlying license is still valid and present.

> **Note --** **Practical consequence.** A customer who uninstalls ForteFide without preserving `license.key` has destroyed the trust anchor. Their previously-shipped evidence ZIPs still exist as files, and the manifest signatures are still cryptographically valid at the instant of issue -- but provenance can no longer be re-derived. Treat such packages as unverifiable for new assessments.

Cryptographic primitives

ITEM	ALGORITHM	VALUE BUNDLED IN PACKAGE
Manifest signature	Ed25519	`manifest['signature']` -- base64-encoded 64-byte signature over canonicalized JSON of the manifest with the signature field removed.
Public key	Ed25519	`manifest['public_key']` -- base64-encoded 32-byte public key. Same key embedded in the ForteFide build that produced the package.
Per-artifact hash	SHA-256	`manifest['artifacts'][i] ['sha256']` -- 64-hex-character digest of the file as included in the ZIP.
Manifest digest	SHA-256	`manifest['manifest_sha256']` -- digest of the canonicalized manifest JSON (informational; the signature is the authoritative integrity check).
Canonicalization	`json.dumps(sort_keys=True)`	All keys sorted; standard JSON whitespace. Guarantees the exact same bytes are signed and verified across implementations.

Step 1 -- run the bundled verifier (recommended)

```
$ unzip ForteFide_Signed_Evidence_42.zip -d ff_evidence
$ cd ff_evidence
$ python3 verify_evidence.py

 OK ForteFide_Scan_Report_42.pdf
 OK 01_System_Security_Plan.pdf
 OK 02_POA_M.pdf
 OK 04_SPRS_Scorecard.pdf
 OK 05_Control_Evidence.pdf
 OK scan_data.json
 OK dd-trust-manifest.json

Signature: VALID (Ed25519, public_key signed in)
```

`verify_evidence.py` requires PyNaCl (`pip install pynacl`). It runs in any Python 3.8+ environment. The script source is plain text in the ZIP; you may inspect it before running.

Step 2 -- manual verification with openssl (alternative)

If you prefer not to run vendor-supplied scripts, you can verify manually. The bundled public key is base64 inside `manifest['public_key']`. openssl 3.0+ is required for raw Ed25519 verification:

```
# 1. Extract the raw 32-byte Ed25519 public key from the manifest, wrap it in a
#    PEM SubjectPublicKeyInfo header, reproduce the exact canonicalized-JSON bytes
#    that were signed (every manifest field except signature/public_key), decode
#    the base64 signature, then:
openssl pkeyutl -verify -pubin -inkey pub.pem \
  -rawin -in payload.bin -sigfile sig.bin
# Expected: 'Signature Verified Successfully'
```

Trust assumptions in signature verification

- The Ed25519 public key bundled in the ZIP is genuine -- cross-check it against the public key shipped in the ForteFide installer. If they match, the package was sealed by a build holding the matching private key.
- The private key has not been compromised -- it stays on the operator workstation; ForteFide does not transmit it.
- The customer has not generated a forged package by modifying ForteFide source -- ForteFide is partially open-source; the cryptographic chain proves the package was produced by something with the matching key, not that no source modification occurred. Treat the signature as integrity evidence and apply independent judgment about source authenticity.

Tip -- Practical bar. Signature verification gives you confidence the package you received is the package the customer's ForteFide instance sealed. It does not, on its own, prove the scan accurately reflects the target's state. For that, combine the package with manual spot-checks of the underlying targets (Section 8, Sample Assessor Workflow).

What to do if verification fails

- **TAMPERED <filename>** -- the artifact's bytes do not match the SHA-256 in the manifest. Do not accept.
- **MISSING <filename>** -- the manifest references a file not present in the ZIP. Request the original ZIP directly from the customer's ForteFide install.

- **Signature: INVALID** -- the Ed25519 signature does not validate against the bundled public key. Do not accept; request a fresh package after investigation.
- **Signature: not present** -- the package may be the unsigned DRAFT variant produced by the free tier. Confirm with the customer; they need a Pro license to produce a signed package for assessor evidence.

6 | Auth Tier Disclosure

ForteFide uses a three-tier authentication cascade for operator-to-target access. The tier used directly affects how much weight a C3PAO should place on per-control evidence. The scan report and remediation log record the tier used per target.

TIER	MECHANISM	STRENGTH	COMPLIANCE POSTURE
Tier 1	SSH certificate (Ed25519 user cert with 30-day validity, issued by a license-derived CA)	Strongest	Preferred. Cert expiry binds the auth window to the ForteFide engagement; no long-lived static credential.
Tier 2	SSH public key (Ed25519 keypair, deployed at Prepare)	Strong	Acceptable. Public key stays in target authorized_keys until Teardown removes it.
Tier 3	Password (admin-supplied, used once for Prepare or as scan-time fallback when SSH key/cert deploy failed)	Weakest	Compliance scrutiny flag. Persistent password use indicates Prepare did not complete cleanly. Verify the customer has a documented reason.

How to read the tier in evidence

Per-control evidence in `05_Control_Evidence.pdf` and `scan_data.json` includes the auth tier used -- look for the `auth_tier` field, or the "Auth: tier-N" line in the per-host header of the scan report.

Note -- Tier 3 is a flag, not a fail. A Tier 3 entry does not invalidate the scan. It indicates the control evidence was gathered over a password-authenticated channel, which is a weaker assertion of "only ForteFide acted on the target" than Tier 1 or Tier 2. Confirm with the customer; do not silently accept.

Lockout-warning controls in the evidence trail

A small set of Linux controls are enabled for automatic remediation only behind an explicit per-control, per-target operator acknowledgment (ACK) -- each carries a non-trivial chance of locking ForteFide out of the host if applied without coordination. The remediation log records the ACK timestamp and operator identity alongside the apply record. Absence of an ACK record for one of these controls means it was never remediated; presence means the operator deliberately accepted the lockout risk before the change ran.

- Evidence to look for: an ACK-timestamp field in the remediation-log entry for any lockout-warning control ID.
- The scored denominator is unchanged by ACK-gated remediation -- the automated compliance percentage is computed over the 79 technically-assessed controls; the 31 organizational controls are excluded from the denominator, not failed.

7 | Manual-Only Controls

Some 800-171 controls cannot be reliably verified by an automated remote scanner. ForteFide flags these as MANUAL and does not attempt to remediate them automatically. Out-of-band assessor verification is required.

Categories ForteFide flags as manual

- **Domain policy controls (DC-aware)** -- a small set of controls are DC-aware: ForteFide detects whether each Windows target is a domain controller (`DomainRole >= 4` via `Win32_ComputerSystem`) and chooses behavior per host -- APPLIED on a DC (the customer is the policy authority for their own AD), SKIPPED on a domain member (AD-pushed policy overrides local).
- **Physical security** -- PE family (Physical and Environmental Protection) controls require on-site verification. ForteFide cannot inspect a server room.
- **Personnel controls** -- PS family (Personnel Security) controls require records-based verification of background screening and termination procedures. ForteFide cannot read HR systems.
- **Awareness training** -- AT family controls require attestation of completed training; ForteFide produces a scaffold with attestation placeholders.
- **Policy documents** -- family-level policy review and approval cannot be content-verified by a scanner. ForteFide produces policy scaffolds; the customer must complete and approve.

How manual-only entries appear in evidence

Manual-only controls in `05_Control_Evidence.pdf` and `scan_data.json` do NOT receive an automated MET. They emit ATTESTATION_REQUIRED (`passed=null` , flagged for attestation) -- ForteFide deliberately does not auto-pass a control it cannot technically verify. Out-of-band evidence the customer supplies (signed policies, training records, physical-security inventory, HR records) goes alongside the ForteFide package.

Manual-control attestation -- in-product, sealed

Current ForteFide builds let the customer attest to the manual-only controls inside the product. The customer signs a per-family attestation PDF; each signed PDF folds into the evidence ZIP under the existing Ed25519 seal, and a sealed `10_Attestation_Index.pdf` records, per family, whether the attestation is signed (ATTESTED) or still pending (ATTESTATION REQUIRED).

The automated compliance percentage is unaffected by attestation: the score is computed over the **79** technically-assessed controls, and the **31** organizational controls are excluded from that denominator. Attestation does not move the automated number; it supplies the customer's signed assertion for the controls a scanner cannot evaluate.

> **Note -- Honesty boundary -- what the seal does and does NOT mean.** The Ed25519 seal on an attestation PDF provides chain-of-custody integrity and provenance ONLY -- it proves the file was assembled by ForteFide at the stated time and has not been altered since. It is explicitly NOT a claim that ForteFide verified the underlying control. Verification that the control is actually implemented is the sole responsibility of the signing official; the customer's signature is the attestation, and a false attestation is the customer's liability, not ForteFide's. Treat a signed attestation as the customer's representation to be examined and interviewed against, not as tool-verified evidence.

What this means for the assessor

- Treat the ForteFide score as a partial result. The automated probes cover the controls a scanner can reliably verify; the manual-only controls require your independent review.
- Cross-check the customer's placeholder fields in the SSP, IRP, and Training Records. Empty placeholders indicate incomplete customer work, not a ForteFide deficiency.
- Confirm DC-aware controls match the customer's deployment shape -- expect APPLIED determinations on a small contractor's sole DC and SKIPPED on domain members.

8 | Leave-No-Trace & State-Change Verification

ForteFide is designed so that, after Teardown, the customer's targets are returned to the state they were in before Prepare. ForteFide also ships paired, cryptographically signed install/remove records for every state change it makes on a target host, so an assessor can verify the leave-no-trace claim offline rather than trusting operator say-so.

What gets removed at Teardown

ITEM CREATED DURING PREPARE/REMEDIATE	REMOVED AT TEARDOWN?
Per-host service account	Yes
Service-account home directory and authorized_keys	Yes (entire account is deleted)
Deployed SSH public key	Yes (with the account)
Deployed SSH user certificate	Yes (with the account); also auto-expires after 30 days
sshd_config modifications	Reverted from rollback bundle
Service-account privileges (sudo entries, group memberships)	Yes
Per-control remediations (package installs, registry edits, audit rules)	Reverted only if the customer explicitly invokes rollback from the dashboard. Teardown does NOT auto-rollback remediations.

What's in the evidence package

- `state_changes.jsonl` -- append-only JSONL of every install and remove record for the engagement.
- Each line is one record: `{version, ts, engagement_id, target, kind, action_type, details, actor, sig}`.
- Action types: `svc_user, sudoers, ssh_key, ssh_cert, ca_pubkey, sshd_config, watchdog`.
- `sig` is an HMAC-SHA256 of the canonical record (sorted JSON keys, no whitespace), keyed by material derived from the customer's license.

Pairing install with remove

Pair key: `(target, action_type, canonical(details))`. For every install record there should be a matching remove record with the same pair key. Any unpaired install is a leave-no-trace audit gap. Any orphan remove (no matching install) usually indicates a pre-existing artifact the customer cleaned up, not a cause for concern.

> **Note -- License retention is required for verification.** The signing-key derivation requires the customer's `license.key` file. If the customer uninstalled ForteFide without backing it up, the derived key is unrecoverable -- signatures on file exist but cannot be verified. Treat such artifacts as UNTRUSTED. DenseDefense does not escrow license.key files and cannot regenerate them. Asking the customer for license.key alongside the evidence package is a normal part of the assessment intake checklist for a ForteFide-generated engagement.

What this assurance buys you (assessor angle)

- ForteFide cannot tamper with records after the fact -- the signing key changes per license rotation, and prior signatures invalidate.
- DenseDefense is not in the verification trust path -- the only requirements are the customer's license file and the FF binary.
- No outbound calls to vendor APIs needed; verification works in cleared environments.
- The customer cannot fake a clean teardown by deleting `state_changes.jsonl` entries -- the file is append-only, fsync-durable, and signature-anchored.

Verified remediation (state-change proof)

After each remediation, ForteFide re-runs the same 800-171A check that originally returned NOT MET -- it does not trust the remediation command's exit code. The outcome is recorded as a per-control state-change pair `{control_id, before, after, verified}` and sealed into the signed evidence package as `state_change_pairs.json`, under the same Ed25519 chain of custody as the scan evidence.

- `verified=true` means the re-run of the original check now satisfies the control. `verified=false` means the remediation ran but the control still does not pass -- treat as a NOT MET the customer attempted and failed to close.
- `state_change_pairs.json` is covered by the evidence-manifest SHA-256 and Ed25519 signature like every other sealed artifact; the bundled verifier flags any post-seal edit.

What leave-no-trace does NOT promise

- Rollback of remediations is opt-in, per-control, customer-initiated. ForteFide does not auto-revert remediations at teardown.
- Original-state restoration depends on rollback bundles existing -- if the customer deletes one before invoking rollback, ForteFide cannot restore that control.
- Teardown removes only what ForteFide installed. Modifications the customer made by hand (outside ForteFide) are not tracked or reversed.

9 | Sample Assessor Workflow

A practical workflow for incorporating a ForteFide signed evidence package into a CMMC Level 2 assessment. Adapt to your C3PAO's process.

Phase 1 -- receive and verify the package (15-30 min)

- Receive the signed evidence ZIP via your standard secure transfer channel; note its SHA-256 at receipt time in your assessment workpapers.
- Extract and run `verify_evidence.py` (or the manual openssl path from Section 5).
- Confirm: signature VALID, every artifact OK, no TAMPERED/MISSING entries.
- Cross-check the bundled public key against the public key in the ForteFide installer the customer used.

Phase 2 -- manifest review (15 min)

- Open `evidence_manifest.json`. Record scan_id, generated_utc, machine.hostname, machine.fingerprint, license.org_id, license.license_id, product_version.
- Confirm license.org_id matches the OSC you are assessing, and generated_utc is within your evidence-currency policy (commonly 30 days).

Phase 3 -- per-control review (variable)

- Open `05_Control_Evidence.pdf`. For each NIST family, spot-check 3-5 controls that received a MET determination. Read the captured command and raw output; confirm the determination follows from the output.
- Note the auth_tier per host. For Tier 3 hosts, increase spot-check density or request a re-Prepare/re-scan.

Phase 4 -- live verification (recommended)

- Pick 3-5 controls across families. With customer consent, run the same probe by hand against the same target and compare to the captured evidence.
- If results diverge, the target has changed since generated_utc, or the evidence is stale -- request a fresh scan and a new signed package.

Phase 5 -- manual-only controls (variable)

- Review controls in the PE, PS, AT families per Section 7.
- Cross-check placeholder fields in the SSP, IRP, and Training Records PDFs. Empty fields are customer deficiencies, not vendor deficiencies.
- Request out-of-band evidence: signed policies, training completion records, physical security walkthrough.

Phase 6 -- SPRS scoring sanity check (10 min)

- Open `04_SPRS_Scorecard.pdf`. Confirm the starting score of 110 and per-severity weights (Critical -5, High -3, Medium -3, Low -1).
- Recompute manually from the failed-control list in `scan_data.json`; confirm the SPRS score in the PDF matches your computation.

Phase 7 -- document the assessor determination

Record in your workpapers: package SHA-256, signature verification result, manifest identifiers, the ForteFide version that produced the package, the auth tiers observed, the controls you spot-checked, the live-verification results, and the manual-only evidence obtained out-of-band. Your assessment determination derives from this composite, not from the ForteFide package alone.

10 | Trust Model & Known Limitations

Stating the trust assumptions and known limitations explicitly so the C3PAO knows what the signature does and does not defend against, and what to plan around.

Threats the signed package defends against

- **Tampering in transit or at rest** -- anyone modifying the ZIP between the customer and the assessor, or after it has sat on a file share, will invalidate the manifest signature and/or an artifact hash.
- **Selective artifact substitution** -- replacing one file in the ZIP with a different file under the same name fails the SHA-256 check on that artifact.
- **Manifest forgery without the private key** -- an attacker would need the Ed25519 private key (held only by the operator workstation) to produce a manifest that verifies.
- **Cross-package replay** -- a package is bound to its scan_id, generated_utc, license, and machine; an attacker cannot lift a signed manifest from one package onto another scan's artifacts.

Threats the signed package does NOT defend against

- **Operator workstation compromise** -- if the workstation running ForteFide is fully compromised, the attacker has the private key and can produce signed packages with any content. This is the same threat model as any tool that signs locally. The C3PAO mitigates by treating the operator workstation as in-scope.
- **Customer source-code modification** -- ForteFide source is partially open. A skilled customer could fork and modify the scanner, build locally, and produce a signed package with a valid signature but fabricated per-control evidence. The C3PAO mitigates with live verification (Section 9, Phase 4).
- **Customer override of a determination** -- a customer can declare a NOT MET control as MET via override with a rationale. This is a documented and auditable feature, not an attack, but the C3PAO must read the rationale and apply judgment.
- **Information freshness** -- the signed package attests to state at generated_utc; nothing in the cryptographic chain prevents the environment from changing immediately after sealing. Evidence-currency policy is the C3PAO's lever here.

Tip -- Net assurance. The signed evidence package gives strong assurance the package is authentic ForteFide output and has not been tampered with end-to-end. It gives medium assurance the ForteFide instance was not modified. It gives moderate assurance the per-control determinations reflect target state. The C3PAO closes the residual gap with live verification and out-of-band evidence for manual-only controls.

Known limitations

- **Operator-workstation hygiene** -- Teardown cleans up target-side artifacts but may leave per-target cert directories, journal entries, and target-profile state on the operator workstation. No effect on the evidence package the C3PAO receives.
- **Single-instance signing key** -- all ForteFide installations of a given build share the same Ed25519 signing public key (embedded at build time), so the public key alone does not distinguish customers from each other. Use `license.license_id` and `license.org_id` in the manifest for customer-distinguishing identity.
- **DD-Trust rescan-verification scope** -- post-remediation rescan verification fires only after explicit Remediate runs, not after every scan. A scan-only package (no remediation) will contain the evidence-seal event but no `state_change_pairs.json` -- absence means "no remediation was attempted," not "remediation occurred and the verifier was bypassed."

A | Appendix A: Glossary

TERM	MEANING IN THIS GUIDE
C3PAO	Certified Third-Party Assessment Organization. Authorized by The Cyber AB to perform CMMC Level 2 certification assessments.
CCA / CCP	Certified CMMC Assessor / Certified CMMC Professional -- individual credentials under the Cyber AB ecosystem.
OSC / OSA	Organization Seeking Certification / Organization Seeking Assessment. The customer being assessed.
CUI	Controlled Unclassified Information -- the data type the CMMC Level 2 assessment is concerned with.
CMMC	Cybersecurity Maturity Model Certification. DoD compliance framework codified in 32 CFR Part 170.
NIST SP 800-171 / 800-171A	NIST publication enumerating 110 security requirements for non-federal systems handling CUI, and its companion publication defining assessment objectives.
SPRS	Supplier Performance Risk System. DoD scoring format used by prime contractors; ForteFide produces an SPRS scorecard.
SSP / POA&M	System Security Plan / Plan of Action & Milestones -- required compliance documents; ForteFide produces scaffolds.
Ed25519	Edwards-curve digital signature algorithm -- 32-byte keys, 64-byte signatures. Used for the manifest signature.
Auth tier	ForteFide-internal classification of the auth method used per target. Tier 1 = SSH cert, Tier 2 = SSH key, Tier 3 = password. 1 is strongest.
DC-aware controls	ForteFide's internal list of controls whose remediation behavior depends on whether the Windows target is a domain controller -- APPLIED on DCs, SKIPPED on domain members.
Leave-no-trace	Design principle: targets return to their pre-Prepare state at Teardown. Customer-controlled remediation persistence is opt-in via per-control rollback bundles.
Manual-only	A control the scanner cannot reliably verify; ForteFide flags it for assessor review rather than fabricating automated evidence.

B | Appendix B: Control Determination Semantics

Every control in `scan_data.json` carries a determination from a small fixed vocabulary. Understanding each determination is essential to reading `05_Control_Evidence.pdf` correctly.

DETERMINATION	FIELD VALUE	WHAT IT MEANS	ASSESSOR ACTION
MET	`passed: true`	ForteFide ran a probe, captured output, and the output satisfies the control's pass criteria.	Spot-check the captured command and raw output.
NOT MET	`passed: false`	The output does NOT satisfy the control's pass criteria.	Confirm a POA&M entry exists with a severity-appropriate target date.
MANUAL	`passed: null`, `determination: "manual"`	Control is in the manual-only category. ForteFide did not attempt automated probing.	Verify out-of-band: signed policy, training record, physical inspection, HR record.
NOT APPLICABLE	`passed: null`, `determination: "na"`	Customer asserted the control does not apply to this environment. Override rationale required.	Read the override rationale and confirm the asserted environmental fact.
ERROR	`passed: null`, `determination: "error"`	Probe execution failed (timeout, auth failure, target unreachable). The control was NOT evaluated.	Treat as missing evidence; request a re-scan of the affected target.
OVERRIDDEN	`passed: true` with an override flag, `determination: "override-met"`	Customer accepted a compensating control or risk and marked the control MET via override, with rationale.	Apply heightened scrutiny -- the underlying probe may have returned NOT MET.

C | Appendix C: NIST 800-171 Family Coverage

The 14 NIST SP 800-171 Rev 2 families and how ForteFide handles each. "Automated" means ForteFide runs probes and produces a MET / NOT MET determination from raw output. "Manual-flagged" means ForteFide marks the control for assessor or customer review and does not fabricate a determination. "Hybrid" means some controls in the family are automated and some are manual.

ID	FAMILY	FORTEFIDE HANDLING
AC	Access Control	Hybrid. Local account, lockout, session, remote-access controls automated. Policy-statement controls manual.
AT	Awareness and Training	Manual-flagged. Attestation-based; ForteFide produces a Training Records scaffold.
AU	Audit and Accountability	Mostly automated. Audit subsystem state, log retention, log-write success captured via probes.
CA	Security Assessment	Hybrid. Self-assessment frequency and POA&M presence automated; system-boundary documentation manual.
CM	Configuration Management	Mostly automated. Baseline drift, software inventory, least-functionality probes run. One control is DC-aware.
IA	Identification and Authentication	Mostly automated. Password policy, MFA presence, account ageing, identifier reuse all probed. One control is DC-aware.
IR	Incident Response	Hybrid. IRP existence and contact accuracy via a scaffold document; tabletop-exercise evidence remains organizational.
MA	Maintenance	Hybrid. Remote-maintenance auth automated; on-site maintenance ledger manual.
MP	Media Protection	Mostly manual-flagged. Removable-media policy, sanitization, transport are organizational.
PE	Physical and Environmental Protection	Manual-only. ForteFide cannot inspect server rooms or verify badge logs.
PS	Personnel Security	Manual-only. Background-screening and termination records are HR-system records ForteFide does not access.
RA	Risk Assessment	Hybrid. Vulnerability-scan presence and remediation cycle automated; risk-register content review remains assessor-side.
SC	System and Communications Protection	Mostly automated. Boundary protection, DNS, encryption-in-transit, secure baselines all probed.
SI	System and Information Integrity	Mostly automated. Patch level, malicious-code defense, monitoring presence all probed.

Total: **110** NIST 800-171 r2 controls. ForteFide technically scans **79** of the 110; **31** are organizational (training, physical, personnel), assessed via attestation and out-of-band documentation. The automated compliance percentage is computed over the 79 technically-assessed controls; the 31 organizational controls are excluded from that denominator, not counted as failures. Automated remediation covers **67** controls on Linux and **66** on Windows (**82** distinct controls across the union of both) -- the remainder are manual-flagged for the assessor's independent verification or are doctrine-skipped on DC targets to prevent self-lockout. Exact per-control automation status is in `scan_data.json` under `control_results[].determination`.
