Gankpo Trust Ecosystem Alignment

Version: 1.0.0 **Status:** Approved **Scope:** Gankpo Suite, Gankpo Reader, Gankpo Reader Mobile, Gankpo Field.

1. SHARED TRUST ENGINE CORE

The core verification of all Gankpo products now relies on a centralized module: `TrustEngineCore` This engine guarantees a perfectly consistent trust ecosystem across all runtimes.

It handles:

- Hash verification (BLAKE3)
- Signature verification (Ed25519)
- Trust anchor validation (Institutional CA, Trust Registry)
- Custody chain validation
- Policy enforcement (Nested/Ordered)
- Share policy validation

2. TRUST ENGINE API

The primary interface is `verify_container()` which returns the `TrustEngineResult` payload.

Mandatory Signature Statuses:

- `Valid`
- `Not Verifiable`
- `Invalid`

3. READER MOBILE ALIGNMENT

Scope: Lightweight verifier.

- Displays: Document preview, Trust summary, Signature status, Timestamp status, Trust score, Proof timeline, Basic forensic metadata.
- **Excluded:** Trust simulation, Proof diff, Advanced forensic inspection.

4. MOBILE UI MODEL

- Trust summary displays clear checkpoints: Integrity ✓, Signature ✓, Issuer ✓, Trust Score 100.
- Actions available: Open document, Timeline, Proof details.

5. FIELD APPLICATION ALIGNMENT

Scope: Rapid verification and active capture.

- Functions: verify document integrity, display signature status, display issuer identity, display trust score.

6. FIELD EVIDENCE CAPTURE

Field operations allow the generation of hardware-anchored Gankpo containers directly on offline devices. Process: capture evidence -> compute BLAKE3 hash -> sign evidence -> create custody event -> attach

metadata -> sync with Suite.

7. FIELD TRUST VIEW

A customized trust view adapted for low-visibility, high-pace environments. Emphasis on:

- **Integrity**
- **Signature**
- **Issuer**
- **Trust Score**

8. SUITE INTERNAL READER ALIGNMENT

The internal Gankpo Suite verification processes must utilize `verify_container()`. Pages updated: `VerifyPage`, `ExpertVerifyView`, `DocumentActionPanel`.

9. SUITE EXPERT INSPECTION

Suite and Desktop Reader expose the **UniversalProofInspector** with all tools unlocked:

- Proof, Timeline, Graph, Forensic, Container, Security, Simulation, Compare.

10. TRUST ENGINE CONSISTENCY

A single truth. An identical container must output the exact same verification result whether evaluated on Suite, Reader, Reader Mobile, or Field.

11. BACKEND TRUST AUTHORITY MODEL

The `gkp_trust_engine` Rust module serves as the exclusive and absolute source of truth for cryptographic validation. All validations (integrity, signatures, anchors, custody chains, policies, scores) are strictly computed by Rust. The frontend `TrustEngineCore` is uniquely a transparent wrapper that invokes the backend (`trust_engine_verify_container`), devoid of any independent deduction capability.

12. SIGNER RESOLUTION ARCHITECTURE

The exact resolution of a signer's identity lies securely within the `SignerResolutionEngine` inside the Rust backend. This ensures an absolute, unguessable display of identities across all systems.

- Every `SignerInfo` object carries a definitive `resolution_status`.
- `unknown_authority` or `missing_identity_binding` are never fallback UI guesses; they are strict cryptographic statuses returned explicitly by Rust when applicable.

13. FAKE READER PROTECTION

To combat forged implementations, a software verification stamp is now required. The application indicates: `Verified by Gankpo Reader` along with:

- Software signature
- Reader version
- Verification engine version

14. GKP IMPLEMENTATION WARNING

If a GKP is processed strictly but the engine does not pass authenticity checks, the following alert is displayed: `Verification engine not recognized.`

15. TRUST ENGINE VERSIONING

Every verification report explicitly embeds:

- `engine_version` (e.g., v1.0.0)
- `container_version` (e.g., v4)
- `engine_signature` (e.g., "GANKPO_OFFICIAL_SIG")
- `is_official_engine` (e.g., true)

16. CROSS PLATFORM TESTS

Comprehensive validation via Vitest integration and mock simulation logic testing the bridging pipeline between Swift/Tauri into to `TrustEngineCore`: `test_signer_resolution_in_rust`, `test_frontend_wrapper_only`, `test_unknown_signer_only_when_resolution_fails`, `test_cross_runtime_same_signer_resolution`, `test_trusted_signer_never_displayed_as_unknown`, `test_engine_signature_protection`.