

PrismManifest - IMPLEMENTATION PLAN

PrismManifest Implementation Plan & Production Benchmarks

Field Value
Project PrismManifest / FinancePackBench
Status Approved Engineering Implementation Plan
Version 2.2.0
Date 2026-08-08 (addendum 2026-08-09)
Author Senior QA / Architect & System Design Au

1. Executive Summary & Production SLAs

This document outlines the step-by-step engineering roadmap to implement, benchmark, and deploy the PrismManifest Architecture.

Measurable Production Engineering SLAs

Production Metric Target Engineering SLA Enforcement Mechanism
False-Accept Rate (Critical Money Fields) $\leq 0.01\%$ Hard BLAKE3 digit matching + Layout Anch
False-Refusal / Escalation Rate $\leq 5.0\%$ High-precision layout parsers minimize u
Audit Replay Reproducibility 100% Bit-Identical Immutable storage of Ed25519 Parameter M
CUDA Parity Rate 100% Gate Agreement Experimental CUDA path (Phase 3) must ma
Gate Verification Latency (Host Path) $\leq 15\text{ ms}$ (p99) In-memory FlatBuffer parsing + C++ numer

2. Quorum Disagreement Thresholds & Status Gate Mapping

Sub-layer 4c implements aligned status gate mappings across all verification paths:

Gate Status Conditions
PASS Disagreement = \$0, OCR confidence ≥ 0.9
PASS_WITH_HUMAN Disagreement $> \$0$ but $\leq \$1,000$, OR OCR
REFUSE Disagreement $> \$1,000$, OR 0% span ground

3. FinancePackBench Benchmark Specification

Corpus Composition (500 Financial Packages)

- * 200 IRS Form 1040 / W-2 / 1099 Tax Packages (2022-2025 multi-year returns).
- * 150 Brokerage Account Statements & Trade Confirmations (complex multi-asset tables).

* 150 Commercial Loan Underwriting Packages (messy scanned PDF balance sheets & income statements).

11 Planted Error Test Classes

- 1. Digit Drop / Addition: \$450,000 mis-extracted as \$45,000.
- 2. Line-Item Anchor Shift: Selecting Line 1 Gross Income instead of Line 11 AGI.
- 3. Temporal Mis-Binding: Selecting 2023 figures instead of 2024 figures.
- 4. Distractor Amounts: Selecting Previous Year Income or Spouse Income.
- 5. Parentheses Negatives: Misinterpreting (10,500) as positive \$10,500.
- 6. OCR Scan Noise: Low-DPI scan causing digit confusion (8 vs 3).
- 7. Prompt Injection (inert): Malicious instructions in the document, but critical param remains evidence-bound -> PASS (trust gate, not injection detector). See docs/FINANCEPACKBENCH_PROMPT_INJECTION_SEMANTICS.md.
- 8. Prompt Injection (followed): Same text, but extractor follows injection -> wrong critical value -> NOT PASS.
- 9. Span-ID Hallucination: Model generating non-existent span pointers.
- 10. Multi-Currency Confusion: Confusing USD and EUR entries in multi-national statements.
- 11. Quorum Disagreement: Discrepancy between LLM extractor and Form Layout Parser.

4. Phased Implementation Roadmap

Weeks Phase
Weeks 1-2 Phase 0: Parameter Manifest Contract & E
Weeks 3-6 Phase 1: Dual-OCR Ingest Indexer, Anchor
Weeks 7-8 Phase 2: Span-Grounded Pointer Schema &
Weeks 9-10 Phase 3: Low-Level CUDA Acceleration & P
Weeks 11-12 Phase 4: Enterprise Compliance Packaging

Phase 0: Manifest Contract & Security Schema (Weeks 1-2)

- * FlatBuffers Schema: Implement binary ParameterManifest schema with Ed25519 attestation fields and dag_compatibility_hash.
- * Canonicalization Library: Write C++/Python numerical canonicalization for fixed-point integer micro-units and parentheses negative handling.
- * Binary Gate Interceptor: Implement the binary gRPC security boundary at the C++ Group 3 endpoint.

Phase 1: Dual-OCR Ingest Indexer & Tool Quorum Gate [CRITICAL PATH] (Weeks 3-6)

- * Dual-OCR Ingest Engine: Implement native PDF text-layer stream + layout-aware neural OCR. Enforce hard OCR confidence floor (≥ 0.98).
- * Form Layout Anchor Parser: Map numeric spans to form bounding boxes and spatial neighbor vectors.
- * Host-Side Quorum Engine (Pattern A) & Plausibility Sub-DAG (Pattern C): Implement deterministic C++ status mapping per §2 (PASS / PASS_WITH_HUMAN / REFUSE).
- * DoD: 0 false-accepts on planted error suite critical money fields; $\leq 0.01\%$ false-accept rate on full FinancePackBench critical money fields.

Phase 2: Span-Grounded Pointer Schema & Line Anchor Resolver (Weeks 7-8)

- * Pointer Schema Transformation (Pattern B): Update tool schemas to expect span_id pointer objects with line-item anchor constraints.
- * Verbatim Span Resolver: Resolve span_id -> Decimal directly from evidence index, enforcing 100% verbatim digit equality.

* DoD: 100% of extracted parameters carry explicit document page, character offsets, and form line-item anchors.

Phase 3: CUDA Acceleration & Parity Testing [Experimental Target] (Weeks 9-10)

* Device Memory Evidence Table: Upload Evidence SoA to VRAM.

* BLAKE3 Fingerprint Search Kernel: Parallel GPU BCD/ASCII digit fingerprint matching.

* CUDA Graphs & TMA: Fuse execution pipeline into static CUDA Graph using TMA async bulk memory transfers.

* Latency Target (experimental): Gate latency p99 adds ≤ 10 ms on GPU path for packages up to 4,000 evidence spans.

Phase 3 CUDA Parity Verification (Weeks 9-10)

* Parity Test Suite: Run the entire FinancePackBench 500-package corpus through both the host-side C++ gate and the experimental CUDA GPU gate.

* Acceptance Criteria: CUDA path must produce 100% identical PASS, PASS_WITH_HUMAN, and REFUSE gate decisions as the host C++ path (and bit-identical Parameter Manifests per System Design CUDA Parity Invariant).

Phase 3 - realized in-tree (2026-08-09)

Item Result
Numba CUDA SoA gate kernel Module-level (prismmanifest/cuda/kernels
Host?CUDA decision parity PASS (run_parity_proof; pytest)
Resident GPU verifier ResidentCudaVerifier - warm p50 ~1.2 ms
Stage profile H->D / kernel / D->H in prismmanifest-ga
Batch throughput 1...1024 fields; CUDA wins at large batc
Anti-pattern fixed Per-request @cuda.jit redefine -> ~115 m
500-pack PyxC++xCUDA parity PASS - bench-manifest-parity (FB roundtr
In-process C++ prismmanifest_c.dll / prismmanifest.cpp_
Still open vs plan True CUDA Graphs capture + TMA (stream-f

Phase 4: Compliance Packaging & FinancePackBench Replay (Weeks 11-12)

* Audit Replay CLI: Tool allowing regulators to replay calculation receipts and highlight exact PDF source bytes bit-identically.

* Red-Teaming & SLA Sign-Off: Validate all 500 FinancePackBench packages against production SLAs, including CUDA Parity Rate when the experimental path is enabled.

Phase 4 - realized in-tree (2026-08-09)

Item Result
Audit replay / escalation Implemented
Compliance red-team + sign-off prismmanifest-gate compliance
Human review UI prismmanifest-gate review-ui
Pilot pack prismmanifest-gate pilot-pack -> reports
Ugly-doc + LLM FA bench-customer-llm (synthetic ugly PDFs;
Arch E2E bench-arch (shared extract -> CPU vs CUD

Implementation Status Snapshot (2026-08-09)

Phase Plan intent Engineering status
0 Manifest + Ed25519 + canonicalize Done (Python + C++)
1 Dual-OCR, anchors, quorum Done (+ tax-year-at-span binding)
2 Span pointers / resolver Done
3 CUDA + parity Decision parity done; Graphs/TMA / full
4 Compliance + bench Pilot packaging done; scale customer PDF

Reproduce: `python -m pytest -q - prismmanifest-gate bench-perf - prismmanifest-gate pilot-pack`.

Document Control

Item Value
Spec ID PRISMMANIFEST-IP-v2.2.0
Classification Approved Engineering Implementation Plan
Supersedes PRISMMANIFEST-IP-v2.1.0
Related Artifacts PRISMMANIFEST_MASTER_SPECIFICATION.md, P