

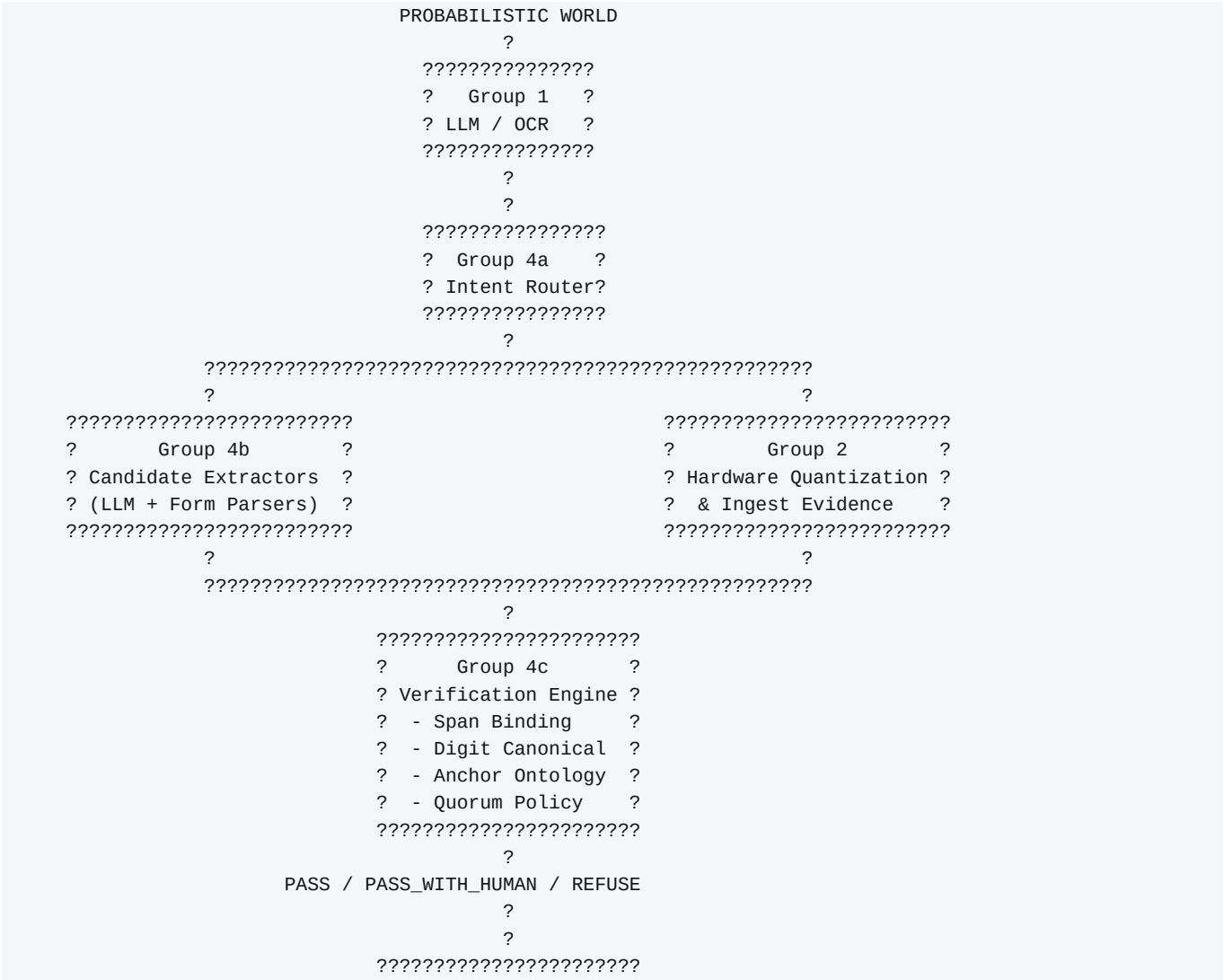
PrismManifest - SYSTEM DESIGN

PrismManifest System Design Specification

Field Value
Project PrismManifest / FinancePackBench
Status Approved Engineering System Design (v2.2)
Version 2.2.0
Date 2026-08-08
Author Senior QA / Architect & System Design Au

1. High-Level Architecture & End-to-End System Diagram

The PrismManifest System enforces an explicit trust boundary between probabilistic LLM interpretation and deterministic software tool execution, resolving the Tool-Argument Trust Gap.



```
?      Group 4d      ?
? Attestor (Ed25519) ?
????????????????????
?
[ VERIFIED PARAMETER MANIFEST ]
?
HARD SECURITY GATE
(Service / Binary)
?
DETERMINISTIC WORLD
?
?
????????????????????
?      Group 3 DAG    ?
? C++ Fixed-Point Math?
????????????????????
```

Functional Group Summary

Group Name Role
Group 1 LLM / OCR Probabilistic perception and interpretat
Group 2 Hardware Quantization & Ingest Evidence Deterministic evidence index / dual-OCR
Group 3 DAG Engine Fixed-point deterministic execution (pos
Group 4a Intent Router Routes work to extractors vs evidence in
Group 4b Candidate Extractors LLM + form parser candidate claims
Group 4c Verification Engine Span binding, canonicalization, anchors,
Group 4d Attestor Ed25519 signed Parameter Manifest issuan

CUDA Parity Policy

CUDA Parity Invariant: The experimental CUDA GPU acceleration path (Phase 3) must produce bit-identical Parameter Manifests and identical gate decisions (PASS, PASS_WITH_HUMAN, REFUSE) as the primary host-side C++ verification engine across the entire FinancePackBench benchmark suite.

The host-side C++ path remains the production authority. CUDA is promoted only when this parity invariant holds and host latency SLOs require acceleration.

Implementation note (2026-08-09): Decision parity host?Numba CUDA is empirically PASS (3/3 core scenarios; pytest green). Production authority for signed manifests remains Python/C++ Ed25519 enforce. Prefer module-level kernels + resident GPU buffers (ResidentCudaVerifier) for warm/batch paths; do not redefine @cuda.jit per request. Single-request C++ CLI latency is dominated by process spawn unless in-process/gRPC.

Implementation Status Addendum (2026-08-09)

Normative contracts above are unchanged. Current engineering proof (see docs/handoff/ and reports/pilot_pack/):

Area Status
Python Group 3 enforce + attestation Production path
C++ FlatBuffer enforce + canonicalize Built; pytest 0 skips
CUDA gate decisions vs host predicates PASS (real GPU)
Warm resident CUDA + batch SoA Implemented; batch=1024 ? CPU host throu
Human review UI / compliance pack Implemented

Ugly-doc + LLM critical FA (synthetic) | FA=0 on pilot pack (8 packs)

Live customer fax corpus | Not yet

2. Quorum Disagreement Thresholds & Status Naming Alignment

Sub-layer 4c applies a deterministic tolerance policy for extractor disagreements, aligned across OCR confidence, form-anchor matches, and quorum votes:

```
[ Quorum Disagreement Policy ]
?
????????????????????????????????????????
? Is Disagreement Amount == $0?      ?
????????????????????????????????????????
?
????????????????????????????????????????
? YES                                ? NO
?                                  ?
[ Check Ingest & Anchors ]      ?????????????????????????????????
?                                ? Is Disagreement <= $1,000?  ?
????????????????????????????????????????????????????????
? All Pass?      ?                                ?
YES?              NO?      ?????????????????????????????????
?                ?        ? YES                                ? NO (> $1,000)
[ PASS ] [ PASS_WITH_HUMAN ] ?                                ?
                        [ PASS_WITH_HUMAN ]      [ REFUSE ]
                        (Escalated to Queue)      (Hard Rejection)
```

Aligned Status Gate Decision Rules

PASS:

- * Exact extractor agreement (Disagreement = \$0).
- * 100% verbatim span-grounding match.
- * Form layout anchors verified.
- * Dual-OCR ingest confidence >= 0.98.

PASS_WITH_HUMAN:

- * Immaterial quorum disagreement (> \$0 but <= \$1,000).
- * OR dual-OCR ingest confidence drops below 0.98.
- * OR layout anchor missing, but span grounding passes.
- * Action: Generates a signed manifest, but routes the transaction to the human review queue.

REFUSE:

- * Material quorum disagreement (> \$1,000).
- * OR span grounding fails (0% verbatim match in corpus).
- * OR Plausibility Sub-DAG order-of-magnitude bounds check fails.
- * Action: Hard rejection. Group 3 execution is blocked.

3. Form Bounding-Box & Line-Item Anchor Ontology

```
{
  "form_type": "IRS_FORM_1040",
  "tax_year": 2024,
  "line_item": "Line_11_AGI",
  "bounding_box": {
    "page": 17,
```

```
    "x0": 120.5, "y0": 450.2, "x1": 240.0, "y1": 465.8
  },
  "spatial_neighbors": {
    "label_left": "Adjusted gross income",
    "label_above": "Line 10: Deductions",
    "line_number_prefix": "11"
  }
}
```

Form Anchor Failure Taxonomy & Mitigations

Failure Mode	Description	Verification	Mitigation Rule
Temporal Mis-Binding	Selecting 2023 AGI instead of 2024 AGI.	tax_year tag on span must strictly match	
Line-Item Shift	Selecting Line 1 Gross Income instead of	Spatial neighbor check must match Adjust	
Entity Ambiguity	Selecting Spouse income instead of Prima	Bounding box spatial region must match P	
OCR Scan Distortion	Low-quality scan produces fuzzy text (\$4	Hard OCR confidence floor (>= 0.98) requ	

4. Attestation Architecture: Ed25519 Digital Signatures

Group 4d issues cryptographically auditable Parameter Manifests signed via Ed25519:

```
{
  "dag_id": "capital_gains_v3",
  "fields": [
    {
      "name": "agi_usd",
      "value_fixed": "450000.00",
      "unit": "USD",
      "provenance": [
        {
          "doc_id": "tax_pack_2025.pdf",
          "page": 17,
          "form_anchor": "Form 1040, Line 11",
          "tax_year": 2024,
          "char_start": 1420,
          "char_end": 1427,
          "raw_text": "450,000",
          "ocr_confidence": 0.99
        }
      ],
      "digit_fingerprint": "8f3a2c4e...",
      "method_votes": ["llm_a", "table_parser", "regex_1040"],
      "verify": {
        "span_match": true,
        "anchor_match": true,
        "digit_match": true,
        "dual_ocr_consensus": true,
        "rule_ok": true
      }
    }
  ],
  "quorum": {
    "required_votes": 2,
    "obtained_votes": 3,
    "disagreements": []
  },
  "gate": "PASS",
```

```

"attestation": {
  "algorithm": "Ed25519",
  "key_id": "kms-key-2026-prod-fin-01",
  "signature_base64": "vM38sK2L8X1gH9mP..."
}
}

```

HMAC Demotion: HMAC-SHA256 is demoted to optional, intra-cluster transport-layer integrity checks.

5. Numerical Canonicalization Rules

Before performing digit fingerprinting or quorum checks, all numerical claims are normalized into canonical representations:

1. Fixed-Point Scaling: Money values are stored as 64-bit integer micro-units (`value_fixed_micro = int(Decimal * 1,000,000)`). \$450,000.00 becomes 450000000000. Manifest JSON may render the human-readable decimal string in `value_fixed`.
2. Currency & Formatting Stripping: Strips currency symbols (\$, €, £), thousands separators (,), and trailing whitespace.
3. Negative Number Normalization: Parentheses negatives (100.50) and trailing minus signs 100.50- are normalized to -100.50.
4. Digit Fingerprinting: `digit_fingerprint = BLAKE3(canonical_ascii_digits)`.

6. Hard Security Boundary: Service & Binary Gating

The Python `@parameter_gated` decorator is a Developer Experience (DX) convenience layer, NOT a security boundary.

The True Security Boundary

The real security gate lives at the Group 3 gRPC / Service Boundary / C++ Binary Entrypoint. The C++ executable enforcing Group 3 calculations strictly validates the Ed25519 signature of the ParameterManifest FlatBuffer payload before opening memory buffers for computation. Any unsigned or unverified invocation fails at the binary interface.

Group 3 accepts execution only for gate statuses PASS and PASS_WITH_HUMAN (the latter only after human approval token / queue clearance per deployment policy). REFUSE never reaches execution.

7. Threat Model & Security Controls

Threat Vector Attack Scenario PrismManifest Security Control
Prompt Injection Document contains malicious text ("Ignor Extractor only emits span_id. Determinis
Span-ID Hallucination LLM invents a non-existent span_id. Kernel 1 / State Machine masks token log
OCR Vendor Collusion Dual-OCR engines share a vendor bug on f Hard OCR confidence floor (>= 0.98) trig
Key Compromise Attestation private key is leaked. Ed25519 Key ID rotation via Hardware Sec

8. Schema & DAG Compatibility Versioning

Every manifest carries explicit versioning headers:

* schema_version: e.g. "2.2.0".

* dag_compatibility_hash: SHA256 hash of the target C++ DAG binary and input schema. If the DAG binary changes, old manifests are automatically rejected.

Document Control

Item Value	
Spec ID	PRISMMANIFEST-SDS-v2.2.0
Classification	Approved Engineering System Design (Fina
Supersedes	PRISMMANIFEST-SDS-v2.1.0
Related Artifacts	PRISMMANIFEST_MASTER_SPECIFICATION.md, P