

PrismManifest - USAGE FOR ENGINEERS AND ARCHITECTS

How Engineers & AI Architects Use PrismManifest

Field Value
Audience Software engineers, AI/ML architects, pl
Package prismmanifest (Apache-2.0)
Companion README, System Design

1. What PrismManifest is (and is not)

PrismManifest is a trust boundary in front of money-sensitive deterministic tools. It is not an LLM, not a full OCR product, and not "prompt-injection defense."

LLM / OCR / parsers -> PrismManifest -> signed ParameterManifest -> Group 3 DAG

Probabilistic systems may propose span pointers. Only a verified, Ed25519-signed ParameterManifest that clears `enforce_group3_boundary` may drive calculators, underwriting engines, or ledgers.

Core equation:

Deterministic Engine + Unverified Probabilistic Input = Deterministic Wrong Answer

2. AI / ML architect placement

1. Treat extractors (Gemini, Azure DI, Tesseract, custom parsers) as untrusted.
 2. Insert PrismManifest after extraction and before any tool/DAG that consumes dollar amounts.
 3. Design for three outcomes:
 - ACCEPT -> run DAG - ACCEPT_PENDING_HUMAN -> review queue - REJECT -> stop (never silently fall back to LLM numbers)
 4. Keep LLMs free to draft narrative; never let model text be the source of money args.
- Invariant: tool arguments for critical money fields come from evidence-bound manifests, not from generative text.
- Optional LLM benches (`bench-llm`, `bench-llm-scale`) measure extractor quality. They do not change the core design of the gate.
-

3. Engineer integration pattern

Happy path

1. Ingest document -> evidence spans (IntentRouter / DualOcrIngestor)
2. Extract Pattern B span pointers (`span_id` + anchors - not raw \$450,000 from the model)
3. `PrismManifestPipeline.run_on_evidence(...)` -> decide gate + sign manifest
4. `enforce_group3_boundary(...)` ? real security boundary
5. Only on ACCEPT, call your DAG

```

from prismmanifest import KeyRing, PrismManifestPipeline, enforce_group3_boundary, GateDecision
from prismmanifest.router import DocumentPackage, IntentRouter

keyring = KeyRing.generate(key_id="local-dev-ed25519")
package = DocumentPackage(
    doc_id="1040.txt",
    pages=["Form 1040 Tax Year 2024\nLine 11 Adjusted gross income: $450,000.00\n"],
    form_type="IRS_FORM_1040",
    tax_year=2024,
)
routed = IntentRouter().run(package)
result = PrismManifestPipeline(keyring).run_on_evidence(
    evidence=routed.evidence, extraction=routed.extraction
)
gate = enforce_group3_boundary(
    result.manifest, public_keys=keyring, expected_dag_id="your_dag_v1"
)
if gate.decision is GateDecision.ACCEPT:
    run_your_dag(gate.manifest) # only here

```

Hardening options

Layer Recommendation
Wire format FlatBuffer (encode_manifest) into C++ /
Latency Prefer Python or in-process prismmanifests
CUDA Optional SoA accelerator; not the sign in
Keys Local for dev; Azure Key Vault envelope
Human review Attach EscalationQueue; clear PASS_WITH_

Do not

- * Treat @parameter_gated as the security boundary (DX only)
- * Skip enforce_group3_boundary because the pipeline "looked good"
- * Feed LLM-printed money strings straight into the DAG
- * Count cuda_sim or SKIP as CUDA parity success

4. Gate model

GateStatus (on the manifest)

Status Meaning
PASS Grounded, OCR >= 0.98, anchors OK, no di
PASS_WITH_HUMAN Immaterial disagreement (<= \$1k), low OC
REFUSE Ungrounded, plausibility failure, or mat

GateDecision (Group 3 boundary)

Decision When
ACCEPT PASS, or cleared PASS_WITH_HUMAN with va
ACCEPT_PENDING_HUMAN Awaiting review
REJECT REFUSE, bad attestation, freshness/repla

5. Day-to-day commands

Job Command
Unit / regression <code>python -m pytest -q</code>
SLA / planted FA <code>prismmanifest-gate bench --packages 500</code>
CUDA require <code>prismmanifest-gate bench --require-cuda</code>
500-pack parity <code>prismmanifest-gate bench-manifest-parity</code>
Customer PDF drop <code>prismmanifest-gate bench-customer-pdf --</code>
Compliance <code>prismmanifest-gate compliance --out repo</code>
Review UI <code>prismmanifest-gate review-ui --queue .es</code>
Pilot evidence <code>prismmanifest-gate pilot-pack --out repo</code>

6. Ops packaging (pilot deploy)

Ops packaging is how you run the gate in a service - not the algorithm:

- * Key material (gen-keys / Azure KV)
- * RBAC at the API edge (prismmanifest.ops.rbac)
- * Idempotency on enforce retries
- * Timeouts + metrics (MetricsRegistry)
- * Append-only audit store + review queue

See handoff/PILOT_DEPLOY.md.

7. Honest readiness

Label When
Pilot-ready Synthetic + adversarial FA=0; Py/C++/CUD
Prod-ready (honest) Live customer fax/scanned corpus passes

Finding customers / redacted document drops matters more for production claims than extra multi-model LLM studies.

8. Spec authority

Doc Role
PRISMMANIFEST_SYSTEM_DESIGN.md Technical contract
PRISMMANIFEST_IMPLEMENTATION_PLAN.md Phases & SLAs
PRISMMANIFEST_MASTER_SPECIFICATION.md Executive index
FINANCEPACKBENCH_PROMPT_INJECTION_SEMANT Why inert injection can still PASS