Appendix A — How to Claim Conformance (MDAB v0.1)

Purpose: minimal, reproducible conformance claim using deterministic hashes and executable tests. This appendix is normative for evidence packaging.

A0. Hash naming (normative)

MDAB v0.1 uses SHA-256. Conformance bundles MUST use ***_hash** naming.
• policy_hash = SHA-256(canon(policy))
• decision_hash is the normative name for the decision binding hash.
Compatibility alias: decision_digest MAY be accepted for ingestion only; if both appear they MUST match; conformance claims MUST reference decision_hash.

A1. Conformance profiles

Profile	Minimum requirements (summary)
MDAB-System-Core-0.1	Authority-before-execution via ATKN; deterministic canon() + decision_hash; Enforcer fail-closed; ev
MDAB-System-Audit-0.1	Core + Verifier replay: chain validation; decision_hash recomputation; policy_hash verification (wher
MDAB-System-HighAssurance-0.1	Audit + signatures and tamper-resistance; WORM/append-only storage; atomic anti-replay for one-ti

A2. Evidence bundle (required artifacts)

A conformance claim MUST include an evidence bundle (repo URL or offline archive) with SHA-256 checksums.

Core boundary artifacts (MDAB)	Paths (examples)
Policy + hash	policy/mdab.core-0.1.0.json; policy/mdab.core-0.1.0.sha256
Golden vectors + CT	vectors/tv01-10.jsonl; ct/CT-INDEX.md; ct/run_ct.py
Negative fixtures (recommended)	fixtures/**; ct/run_ct_neg.py
EER fixtures (recommended)	fixtures/eer/*.jsonl; ct/run_vfy_ct.py

Telemetry CTS artifacts (MDAB-TEL v0.1)	Paths (examples)
Terminology + Schemas	SPEC/terminology.md; schemas/*.schema.json (+ HA overlays)
Requirements + Traceability	ct/REQUIREMENTS.md; ct/TRACEABILITY.md
Schema validator + chain verifier	ct/validate_schema.py; ct/run_chain_verifier.py
Tamper fixtures + signed fixtures (HA)	fixtures/tel/*.jsonl; keys/*

A3. Reference procedure (how to produce evidence)

Run commands offline and capture stdout verbatim in RESULTS.md. Any failure is NONCONFORMANT.

Commands (MDAB core):
`python3 ct/run_ct.py`
`python3 ct/run_ct_neg.py` (recommended)
`python3 ct/run_vfy_ct.py --profile audit fixtures/eer/eer-stream-valid.jsonl` (Audit+)

Commands (Telemetry CTS):
`python3 ct/validate_schema.py`
`python3 ct/run_chain_verifier.py --profile audit fixtures/tel/tel-stream-valid.jsonl`
`python3 ct/run_chain_verifier.py --profile ha fixtures/tel/tel-stream-signed-valid.jsonl` (HighAssurance)

Golden policy_hash: `f2d4239831f71aad992a7a21c6f9e87627fe39d387bf201ab633c511d3c635d3`

A4. Minimal conformance claim (template)

• Implementation: <name> / <version> (commit optional)
• Claim date: <YYYY-MM-DD>
• Claimed profiles: MDAB-System-{Core|Audit|HighAssurance}-0.1
• policy_hash: (golden) + policy reference path
• Vectors: vectors/tv01-10.jsonl (TV01..TV10)
• Test execution: ct/run_ct.py -> PASS (+ optional runners per profile)
• Evidence bundle: repo/archive + SHA256SUMS.txt + RESULTS.md

A5. Packaging checklist

Include: (1) SHA256SUMS.txt for every artifact, (2) RESULTS.md, (3) MANIFEST.md, (4) optional architecture diagram. Avoid external dependencies; reviewers should reproduce results offline.