Step-by-Step Implementation Guide

Intelligence Tokens (ITs) + Object Store / Vector Store Prototype (Template for ASCE Leadership B

Build date: 2025-12-24

Prepared for: Eva Lerner-Lam (internal prototyping; not official ASCE policy)

1. What you are building

You are building a prototype workflow that packages domain recommendations into an Intelligence Token (IT) that:
(1) references authoritative, immutable artifacts in an Object Store, and
(2) makes those artifacts discoverable via a Vector Store for retrieval-augmented responses,
while enforcing governance policy gates and preserving attribution metadata suitable for compensation routing.

Key idea:
- The Object Store is the source of truth (canonical artifacts + hashes).
- The Vector Store is the derived index (embeddings + metadata + references back to the Object Store).

2. Minimum architecture (conceptual)

Components
A) Authoring workstation + repository
B) Object Store (prototype: GitHub repository with immutable history)
C) Packetization layer (canonicalization + hashing + manifests)
D) Vector Store (any vendor that supports metadata filtering)
E) Runtime policy gates (filters + human review hooks)
F) Audit log / usage ledger (for attribution and compensation evidence)

Data flow
Objects → Manifest → IT object → Chunk + Embed → Vector records → Retrieval with policy gates → Response with attribution

3. Step-by-step build plan

Step 1 — Stand up the Object Store (prototype)
- Create a GitHub repository that will host /object_store and /schemas.
- Use the folder structure provided in 03_object_store_template/.

Step 2 — Define canonicalization and hashing
- Choose one canonicalization method (e.g., sorted keys + minified JSON).
- Compute SHA-256 hashes of canonical content bytes.
- Treat the hash output as the content-addressed identifier (CID) for prototypes.

Step 3 — Create an Object Manifest
- For each object you want searchable, list:
  object_id, title, type, provenance, content_hash, and repository path.
- Save it using 02_schemas/object_manifest.schema.json.

Step 4 — Create the Intelligence Token (IT) object
- Set token_type = transferable_knowledge for the recommendations package.
- Reference one or more manifest IDs.
- Specify governance tier and required attribution fields.
- Define policy gates (access/licensing/quality).

Step 5 — Convert objects into indexable text
- Extract canonical text from each object (retain originals in Object Store).
- Preserve headings/section labels to maintain governance context.

Step 6 — Chunk and embed
- Chunk at paragraph/section level; keep overlap.
- Generate embeddings using one stable embedding model.
- Produce vector records per 02_schemas/vector_record.schema.json.

Step 7 — Ingest into your Vector Store
- Upsert vector records; ensure metadata filtering is enabled.
- Store policy gate references in each record's metadata.

Step 8 — Implement retrieval with policy gates
- Apply gates before similarity search when possible (pre-filter).
- Otherwise filter results after retrieval, before responding.
- Always return attribution fields with any retrieved content.

Step 9 — Verification & Validation (V&V)
- Run schema validation and hash verification in CI (continuous integration).
- Confirm provenance completeness.
- Test retrieval filters and audit logging.

Step 10 — Demonstrate to ASCE stakeholders
- Use a small set of objects representing governance, ethics, policy, and technical guidance.
- Show: traceability (prompt → retrieved chunks → object IDs → hashes → provenance).

4. Verification & Validation checklist (minimum)

Critical checks
- Schema validation passes for all manifests, IT objects, and vector records.
- Hash verification: recomputed SHA-256 matches recorded hashes.
- Provenance: each object contains provenance, author/org, publication date (if known), and license/rights.
- Policy gates: retrieval enforces access and licensing constraints.
- Attribution: responses include citation-ready attribution metadata.
- Audit logging: record prompt context, retrieved record IDs, and policy gate decisions (without storing sensitive data).

5. Compensation and attribution (prototype-ready, policy-neutral)

This prototype does not prescribe ASCE policy. It supplies metadata and logging hooks that could enable compensation programs.

Recommended evidence artifacts
- Usage log entry: token_id, object_id, chunk_id, timestamp, user role, gate decision.
- Attribution bundle: author, organization, publication date, license, and canonical object hash.

Human review requirement
- Any compensation determination should be gated as “require_human_review” until an approved program exists.