Local-First Dynamic Persona Intelligence System (LDPIS)

Abstract

LDPIS defines a local-first, air-gapped architecture for transforming large, heterogeneous corpora into grounded, attributable, and conversationally explorable intelligence using generative models. The system operates entirely offline and introduces a deterministic, model-agnostic **persona lens** that governs interpretation, reasoning, and output generation. The persona lens evolves through controlled, auditable updates derived from input heuristics, enabling long-term adaptation without loss of reproducibility. This specification formalizes system components, data structures, execution order, and constraints such that independent implementations can be constructed without reliance on proprietary services or external connectivity.

---

1. Introduction

Generative language models are increasingly used to synthesize knowledge from large corpora; however, prevailing approaches rely on cloud-based infrastructure, opaque reasoning processes, and probabilistic generation that can result in hallucinations and loss of provenance. LDPIS addresses these limitations by defining a system in which generative models function as constrained reasoning engines operating through explicit, evolving lenses of human intent.

The system is designed for environments where privacy, security, cost predictability, and auditability are critical. Examples include research, journalism, regulated industries, and high-security deployments.

---

2. Terminology

- **CKU (Canonical Knowledge Unit)**: A normalized, attributable representation of ingested data.
- **Persona Lens**: A structured, weighted parameter set governing model behavior.
- **Heuristic**: An explicit signal derived from input used to guide reasoning or persona updates.

- **Deterministic**: Producing identical outputs given identical inputs, persona state, and model configuration.

---

3. Design Goals (Normative)

An LDPIS implementation MUST:

- Operate fully offline at runtime
- Support multimodal data ingestion
- Preserve provenance and attribution
- Minimize hallucinations via structural constraint
- Separate persona identity from model weights
- Allow bounded, auditable persona evolution

---

4. System Architecture

An LDPIS implementation SHALL consist of the following components:

1. Ingestion Engine
2. Normalization Pipeline
3. Knowledge Store
4. Persona Manager
5. Reasoning Pipeline
6. Generation Engine
7. Evaluation and Grounding Module
8. Persistence Layer

Components MAY be co-located or distributed within a local environment but MUST adhere to the interfaces and execution order defined herein.

---

5. Data Ingestion and Normalization

5.1 Ingestion Engine

The Ingestion Engine MUST accept one or more of the following modalities:

- Text
- Audio
- Image
- Video

All ingestion and preprocessing MUST occur locally. Implementations MAY use any local model or heuristic extraction method.

5.2 Canonical Knowledge Unit (CKU)

All ingested content MUST be converted into CKUs. Each CKU SHALL contain the following REQUIRED fields:

- id: globally unique identifier
- content: normalized textual representation
- modality: {text | audio | image | video}
- source: original artifact identifier
- provenance: timestamp, extraction method, checksum
- segments (OPTIONAL): sub-units with offsets

Implementations MAY extend CKUs but MUST NOT omit REQUIRED fields.

---

6. Knowledge Store

6.1 Storage Requirements

The Knowledge Store MUST:

- Persist CKUs
- Represent explicit relationships between CKUs
- Support constraint-based traversal

Graph-based storage is RECOMMENDED. Vector embeddings MAY be used as secondary indices but MUST NOT be the sole retrieval mechanism.

6.2 Relationship Types

Implementations SHOULD support at least the following relationships:

- DERIVED_FROM
- REFERENCES

- TEMPORAL_PRECEDES
- SEMANTIC_RELATED

---

7. Persona Lens Specification

7.1 Persona Definition

A persona lens SHALL be defined as:

$P = \{ (a_1, w_1), (a_2, w_2), \ldots, (a_n, w_n) \}$

Where a_i is an attribute and $w_i \in [0,1]$ is its normalized weight.

7.2 Persona File Format

Personas MUST be serialized in a structured format (e.g., JSON, YAML) containing:

- attributes: list of {name, weight}
- constraints: context inclusion/exclusion rules
- system_prompts: ordered prompt fragments
- protocols: reasoning and generation rules

7.3 Model Independence

Persona lenses MUST be model-agnostic. Given identical persona state and context, different compliant models SHOULD produce behaviorally consistent output.

---

8. Reasoning Pipeline (Normative)

For each query, the system MUST execute the following passes in order:

1. Query Parsing
2. Heuristic Extraction
3. Persona Update (OPTIONAL)
4. Context Assembly via Knowledge Traversal
5. Persona-Conditioned Generation
6. Grounding and Attribution Verification

Reordering or omitting passes violates conformance.

9. Persona Evolution

9.1 Update Rule

Persona updates MUST follow a bounded update function:

$P(t+1) = \text{clamp}(P(t) + \Delta P(H), 0, 1)$

Where $\Delta P(H)$ is derived from heuristics extracted from new input.

9.2 Auditability

All persona updates MUST be logged with:

- Timestamp
- Triggering input
- Attributes modified
- Previous and new values

---

10. Generation Engine

The Generation Engine MAY use any local generative model.

It MUST:

- Accept structured context
- Apply persona constraints deterministically
- Disable or seed randomness

---

11. Hallucination Controls

The system MUST NOT generate ungrounded claims unless explicitly configured to do so. Any ungrounded output MUST be clearly labeled.

---

12. Persistence and State

The system MUST persist:

- CKUs
- Knowledge relationships
- Persona files
- Persona evolution logs

---

13. Security Considerations

Because LDPIS operates entirely offline, data exfiltration risk is minimized. Implementations SHOULD protect stored personas and knowledge stores from unauthorized modification.

---

14. Non-Goals

This specification does NOT require:

- Autonomous agency
- Internet connectivity
- Continuous online learning
- Model fine-tuning

---

15. Conformance

An implementation is conformant if it:

- Implements all REQUIRED components
- Adheres to normative execution order
- Preserves determinism
- Operates fully offline

---

16. Conclusion

LDPIS formalizes a new class of generative intelligence systems in which language models function as constrained reasoning engines rather than autonomous oracles. By separating data,

identity, and model, and by enforcing deterministic, auditable execution, the architecture enables trustworthy intelligence synthesis suitable for high-integrity environments.