EARMARK

MACHINES FOR THINKING

PROTOCOL INTRODUCTION

FIRST-ORDER ORIENTATION FOR HUMAN READERS

EARMARK OPEN INTELLIGENCE PROTOCOL
AUTHOR: MIKHAIL SHAKHNAZAROV
BERLIN, FEBRUARY 2026

WHAT THIS IS // 01

This document introduces the Earmark Open Intelligence Protocol: a set of conventions for building and maintaining structured bodies of text -- called corpora -- that govern how language models behave when producing written output. The protocol exists because language models generate fluently, forget immediately, and drift by default. Without external structure, their output is generic, inconsistent, and unverifiable. The protocol provides that external structure.

A governed corpus is not a chatbot configuration or a set of prompts. It is a maintained body of instructions, templates, procedures, and examples that functions as a language specification. The language model executes as a runtime -- a compiler that transforms text under constraints. The intelligence is in the specification, not in the runtime. Vendors are replaceable; the corpus is durable.

THE PROBLEM // 02

Language models produce text that sounds authoritative but carries no provenance, no verification, and no governance. The same prompt can produce different outputs across runtimes, across sessions, and across model versions. There is no way to know whether a given output followed the operator's constraints or drifted into the model's default behavior. There is no way to tell whether a piece of text has been modified since it was produced, or whether it belongs to the category it claims to belong to.

The result is that language model output is disposable. It cannot be trusted, accumulated, or built upon. Every session starts from scratch. Every output requires manual review against unstated criteria. The tool is powerful but the power dissipates because nothing persists.

THE SOLUTION // 03

The protocol addresses this through six structural obligations that a governed corpus must satisfy to be portable across runtimes and durable across time. These obligations are: second-order tokens that carry their own compilation instructions; an artifact coordinate system for routing metadata; compaction transforms for managing high-entropy material; epistemic governance conventions that distinguish settled from provisional content; intrinsic signage that embeds verification patterns directly into prose; and cross-runtime verification that tests portability operationally.

Together, these obligations make it possible to build a body of governed text that maintains its integrity as it moves between tools, between sessions, and between model versions. The text carries its own constraints, its own verification, and its own epistemic status. The operator retains sovereignty over what becomes binding, what remains provisional, and what gets discarded.

THE COMPILER ANALOGY // 04

The core engineering analogy is not "AI assistance." It is compilation. A useful stack view runs bottom-up: physical state transitions; machine code; assembly; high-level languages; and then governed natural language. Each layer increases semantic density per token and expressive power, but also increases ambiguity and the need for governance. Python needs types, linters, and tests. A natural-language medium needs a corpus, explicit procedures, and termination.

Treat every durable instruction as part of the spec. Treat every output as a compiled artifact with a declared audience and verification mode. Treat model upgrades as runtime swaps that require regression checks, because behavior can change without any change in the spec.

WHO THIS IS FOR // 05

The protocol serves anyone who needs language model output to be reliable, verifiable, and accumulative rather than disposable. This includes individual practitioners maintaining personal knowledge systems, teams that need consistent governed output across members and tools, and developers building applications that require structured, verifiable text generation.

The protocol does not require specialized technical knowledge beyond literacy and the willingness to maintain structured text. The skill is teachable. The tools are available. The barrier is conceptual, not technical: the belief that structural thinking about one's own knowledge requires institutional support. It does not. The public library, not the scholar's study, is the historical parallel.

HOW TO READ THIS CORPUS // 06

Start here, with this document. Proceed to the Structural Obligations specification, which defines the six requirements a governed corpus must satisfy. The Coordinate System, Epistemic Governance, Intrinsic Signage, Terse Style, and Design Language documents define the individual protocol components in detail. The Explained Runtime document provides instructions for configuring a language model to interpret (but not modify) protocol-governed artifacts. The Corpus Manifest indexes everything and carries the license declaration.

The protocol documents are self-contained: each specifies its domain without requiring the others to be understood first. Cross-references are explicit. The reading order above is recommended but not required.