Event–Invariant Language (EIL)

Public Specification v1.3.1

EIL is a lightweight, intent-level intermediate representation for preserving meaning, constraints, and invariants across time, paraphrase, and execution boundaries.

Status

This is a clarification release. No breaking changes to grammar or core semantics are introduced.

Core Grammar (Unchanged)

⟦event⟧, ⟦agent⟧, ⟦object⟧ (optional), ⟦intent⟧, ⟦invariants⟧, ⟦constraints⟧ (optional), ⟦time⟧ (optional), ⟦closure⟧.

Intended Use Modes (Normative)

EIL blocks MUST declare or imply their intended use. The following modes are recognized:

- Alignment: establish shared understanding and scope.
- Review: enable validation that invariants are preserved.
- Execution Handoff: provide sufficient invariant detail for direct implementation.

Procedural Content and Invariants

EIL is not a programming language. However, EIL MAY include procedural detail when that procedure itself encodes non-negotiable invariants.

Procedural detail SHOULD be included inline when:
- Multiple implementations would not be equivalent
- Safety, termination, or bounds behavior is invariant
- Error mappings are contractually fixed

Procedural detail SHOULD be external when:
- Multiple algorithms satisfy the same invariants
- Performance tradeoffs are implementation-defined

Permissible Extensions

EIL allows explicitly scoped extensions beyond the core grammar.

Extensions MUST:
- Not contradict declared invariants
- Be explicitly scoped (e.g. implementation-bound, illustrative)
- Not claim universality beyond the declared use mode

Examples of permissible extensions include labeled sections such as Implementation Notes, Acceptance Criteria, or Test Cases.

Procedural Invariant Rule (Key Clarification)

If a procedure is the only valid realization of an invariant, that procedure MAY be treated as invariant-bearing and included directly within an EIL artifact.

Non-Goals (Reaffirmed)

• Universal lossless inter-model language
• Replacement for full algorithm specifications
• Replacement for natural language explanation
• Mandating internal AI representations

Changelog

v1.3:
• Clarified intended use modes
• Defined procedural invariants
• Explicitly permitted scoped extensions
• Resolved ambiguity between intent-only and execution-handoff usage

v1.2:
• Added misunderstanding classes and checklist
• Added intended-use guidance

v1.1:
• Added acknowledgment protocol guidance

v1.0:
• Initial public specification