# Event–Invariant Language (EIL)

## Public Specification v1.3.2

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

This release clarifies best practices for permissible extensions. Non-core fields (extensions) SHOULD be explicitly scoped as local to the artifact and treated as interpretive aids unless elevated into ⟦invariants⟧ or ⟦constraints⟧. No grammar or semantic changes are introduced.

## Version History

### v1.3.2 — Extension Scoping Clarification

Clarified extension scoping as local; no grammar or semantic changes.

### v1.3.1 — Verification Clarification

Introduced verification usage mode and reverse prompts.

### v1.3 — Procedural Invariant Rule

Formalized when procedures may be treated as invariant-bearing.

### v1.2 — Misunderstanding Classes

Documented five common misunderstanding classes.

### v1.1 — Receiver Semantics

Clarified acknowledgment and receiver expectations.

### v1.0 — Initial Specification

Defined core grammar, intent, and non-goals.