FINAL FIOLET ENGINE

Safety Kernel as a Standard

Version: 1.0 (normative)

---

1. Scope

This document defines a **normative standard** for a pre-semantic AI safety kernel.

The kernel is a **deterministic, fail-closed interlock** that decides, *before any content generation*, whether execution may continue.

This standard applies to **all implementations** (software or hardware) claiming compatibility with the FINAL FIOLET safety model.

---

2. Normative Decision Model

The kernel MUST implement exactly one binary decision:

```
CONTINUE | ATOMIC_HALT
```

The decision is made **prior to text generation**, based solely on internal system signals (e.g. deviation metrics), not semantic content.

---

3. Formal Specification (Authoritative)

The authoritative behavioral definition of the kernel is the formal TLA+ specification:

```
SafetyKernel.tla
```

This specification is the **source of truth**. Any implementation behavior not permitted by the formal model is **non-compliant**.

---

4. Core Safety Invariants (Normative)

All compliant implementations MUST satisfy the following invariants:

I1 — Monotonic Halt

Once the kernel enters the halted state, it MUST remain halted forever.

I2 — Single Transition

The only allowed state transition is:

```
Running → Halted
```

I3 — Halt Dominance

If the kernel is halted, all future evaluations MUST return `ATOMIC_HALT`.

I4 — Threshold Trigger

If the monitored deviation exceeds the configured limit, the kernel MUST: - immediately return `ATOMIC_HALT` - irreversibly latch into the halted state

I5 — Determinism

The kernel decision MUST be a pure function of: - current kernel state - current deviation signal - static configuration

I6 — Fail-Closed Behavior

Any execution failure, panic, or undefined condition MUST result in halting behavior.

---

5. Explicit Non-Goals (Normative)

The safety kernel:

- MUST NOT analyze or interpret semantic content
- MUST NOT use machine learning or statistics
- MUST NOT guarantee progress or liveness
- MUST NOT expose any reset or override mechanism
- MUST NOT depend on time, randomness, or external state

---

6. ABI Contract

The canonical C ABI contract is defined in:

```
fiolet_core.h
```

This header defines: - stable binary interfaces - ownership and mutability rules - the absence of any reset or recovery API

Any ABI deviation is **non-compliant**.

---

7. Implementation Notes (Non-Normative)

A reference implementation is provided in Rust (`fiolet-core`), using:

- `no_std`
- no allocation
- `panic = abort`
- explicit FFI boundary

Other implementations (C, C++, FPGA, ASIC) are permitted, provided they satisfy all normative sections.

---

8. Conformance

An implementation is considered **FINAL FIOLET–compliant** if and only if:

1. Its behavior is permitted by `SafetyKernel.tla`
2. It satisfies all invariants in Section 4
3. It exposes no prohibited capabilities listed in Section 5

---

9. Design Philosophy (Informative)

Safety is treated as a property of **system dynamics**, not language.

By removing semantics from the safety boundary, the kernel achieves: - auditability - determinism - formal verifiability

This design aligns with industrial safety interlocks and kernel-level fault containment.

END OF STANDARD