OSC-K Kernel Lint Specification v0.3

1. Purpose

This specification defines a lint/check that verifies an Operational Semantics Contract (OSC) contains the minimum semantic kernel required to function as a contract authority artifact.

The lint evaluates semantic skeleton presence, explicitness, and concrete instantiation. It does not evaluate domain correctness.

v0.3 supersedes v0.2. Changes are motivated by empirical failure modes observed across 30+ contract authoring sessions in production systems (Apr-May 2026).

---

2. Scope

The OSC-K lint applies to OSC artifacts intended to serve as any of:

- Implementation guide input (for derivation)
- Test oracle (for conformance evaluation)
- Safety gate authority artifact (for refusal at commit boundaries)
- Design intent declaration (for governing future implementation)

The lint does not require a formal grammar and does not transform OSC into a programming language.

---

3. Inputs

The lint accepts:

- `contract_text` (required)
- `contract_id` (optional)
- `contract_version` (optional)
- `authoring_mode` (optional, default: `normative`) — see §4.6

---

4. Kernel Requirements (OSC-K)

A contract satisfies OSC-K iff it explicitly provides all six kernel elements:

4.1 Decision Surface

The contract MUST define which outcomes are decision surfaces: outcomes whose interpretation changes externally observable behavior.

The contract MUST include at least one concrete DS category, drawn from:

- halt/proceed (refusal)
- accept/reject
- create/delete
- overwrite/retain
- success/failure classification
- scatter/withhold (pre-validation gates)

The contract MUST name at least one decision surface with system-specific context (entity type, stage name, queue name, or equivalent). Abstract categories alone do not satisfy this element.

4.2 Authority Placement

The contract MUST state where truth lives for deciding desired state.

The contract MUST state at least one non-authoritative category if relevant (e.g., coordination, telemetry, cache, queue), OR explicitly state that no non-authoritative stores/channels exist within scope.

The contract MUST declare **authority directionality**:

- **Normative** (prescriptive): the contract defines what MUST be true. Code that diverges is defective.
- **Descriptive** (extracted): the contract describes what IS true. Code is the authority.

If directionality is not declared, the lint MUST emit WARN with required_fix: "Declare whether this contract is normative (code must match contract) or descriptive (contract must match code)."

Rationale: Ambiguous directionality caused attack-review tools to revert correct architectural fixes by treating code as authoritative over the contract. Explicit declaration prevents this inversion.

4.3 Failure Model

The contract MUST name failure classes that affect DS outcomes.

The contract MUST distinguish at least two failure causes (e.g., policy vs operational vs representation, or equivalent).

The contract MUST distinguish **deterministic** failures (guaranteed to recur on retry with same input) from **transient** failures (may succeed on retry).

The contract MUST state at least one irreversible side effect that is forbidden on failure (e.g., no delete, no publish, no commit, no ack).

The contract MUST state at least one **enforcement action** for each failure class. Valid enforcement actions are:

- halt (stop processing, surface error)
- skip-and-record (mark entity as terminal, continue batch)
- retry (with stated bound)

Enforcement actions MUST NOT include "log and continue" — this is decorative, not enforcement.

Rationale: Systems with catch-and-log validators passed v0.2 while preventing nothing. "Log and continue" is indistinguishable from "no enforcement" in production.

4.4 Identity Semantics

The contract MUST define canonical identity for the primary governed entities.

The contract MUST state any normalization and uniqueness expectations necessary for determinism.

4.5 Operational Bounds

The contract MUST declare at least one bounded execution constraint from:

- Maximum concurrency against a named external dependency
- Backpressure mechanism (queue depth, rate limit, circuit breaker)
- Pre-validation gate (what must be true before bulk execution begins)
- Completion accounting (how success/failure is tracked per-entity, not sampled)

The contract MUST state what happens when a bound is exceeded (the enforcement action — not just the limit).

The contract MUST NOT specify bounds as user-supplied runtime parameters without a system-enforced maximum. A bound that can be overridden at invocation time is not a bound.

Rationale: The most catastrophic production failures occurred when systems had no internal governors — concurrency was user-supplied, pre-validation was absent, completion was sampled not measured. Every safety mechanism was external (a human watching). Systems that pass v0.2 while requiring human-as-control-loop are ungoverned.

4.6 Authoring Mode Declaration

The contract MUST declare its authoring mode at document top:

- **Normative**: This contract prescribes what MUST be true. Code divergence is a defect in the code.
- **Descriptive**: This contract describes current system behavior. Code is authoritative.
- **Transitional**: This contract prescribes target state. Some claims are aspirational (marked). Code is being driven toward this state.

The authoring mode declaration MUST appear before any substantive content (after title/version only).

If a contract is **normative** or **transitional**, findings from automated analysis (SOTEP, attack review, or equivalent) that conflict with the contract MUST be classified as code defects, not contract defects. The contract is the authority.

If a contract is **descriptive**, findings that show code-contract divergence MUST be classified as contract staleness.

Rationale: SOTEP (source-of-truth extraction from code) repeatedly promoted unauthorized implementations and accumulated hacks into governing contracts. Attack reviews then defended those hacks against the author's architectural corrections. Explicit authoring mode prevents authority inversion.

---

5. Precision & Scope Rule

The OSC-K lint is a minimum semantic skeleton check.

The lint MUST NOT penalize contracts for additional detail, verbosity, or precision beyond the kernel.

Additional clauses, definitions, examples, domain-specific requirements, and implementation notes are permitted.

The lint MAY emit WARN or BLOCK only if additional text:

- contradicts a kernel declaration, OR
- makes a kernel element ambiguous, OR
- omits a kernel element entirely.

---

6. Concrete Instantiation Rule

A kernel element is satisfied only if its content is **system-specific**. Generic restatements of the kernel requirement do not satisfy it.

The lint MUST apply the **Instantiation Test** to each kernel element:

> Could this text appear unchanged in a contract governing a completely different system?

If yes → the element is abstract/generic → status: WARN with required_fix: "Instantiate with system-specific facts (names, values, identifiers)."

Examples of FAILING the instantiation test:

- "Truth lives in the authoritative data store" (which store?)
- "Failures are classified as transient or permanent" (which specific failures? what causes them?)
- "The system has bounded concurrency" (what bound? against what? enforced how?)

Examples of PASSING:

- "Truth lives in `CSL/offering.json` (S3). Qdrant is a derived query index, rebuildable."
- "NoBifError is deterministic (entity lacks video content). MAPITimeout is transient."
- "Hydration Lambda concurrency: 15 max (Terraform-enforced). MAPI cannot sustain > 30 concurrent."

Rationale: Contracts filled with abstract restatements of OSC-K requirements passed v0.2 while having zero operational authority. Engineers found these documents "worthless" because they contained no actionable information.

---

7. Document Structure Guidance

OSC-K mandates the PRESENCE of kernel elements, not their POSITION in the document. This section is advisory, not normative for the lint.

The following guidance reflects observed authoring failures:

- Kernel elements may appear in any section order. Optimize for reader navigation, not spec compliance.
- Domain-specific section names are preferred over OSC-K vocabulary. ("Face Processing Overview" over "Entry Point"; "Source of Truth" over "Authority Placement".)
- Schemas belong adjacent to the stage/boundary they govern, not in a monolithic dump section.
- Critical constraints (prohibitions, preconditions) must be visually prominent, not buried in prose.
- A glossary of named concepts (abbreviations, canonical references) should precede first use.
- Numbered sections with a table of contents enable jump-to navigation.
- Historical context, implementation notes, and defensive justifications are not contract material. They belong in commit messages, ADRs, or appendices — not normative sections.

---

8. Evaluation Outcomes

The lint MUST output exactly one of:

- **PASS**: all six kernel elements present, explicit, and concretely instantiated
- **WARN**: all six present but one or more are ambiguous, vague, or fail the Instantiation Test (§6)
- **BLOCK**: one or more kernel elements missing, internally contradictory, or specify "log and continue" as an enforcement action

---

9. Report Format (Normative)

The lint MUST output a structured report containing:

- `result`: PASS | WARN | BLOCK
- `findings`: array of finding objects

Each finding MUST contain:

- `kernel_element`: DECISION_SURFACE | AUTHORITY_PLACEMENT | FAILURE_MODEL | IDENTITY_SEMANTICS | OPERATIONAL_BOUNDS | AUTHORING_MODE
- `status`: OK | WARN | MISSING | CONTRADICTORY | ABSTRACT
- `evidence`: excerpt(s) from contract_text sufficient to justify the status
- `required_fix`: directive text describing what must be added or clarified

---

10. Required Fix Prompts (Templates)

If a kernel element is MISSING, WARN, or ABSTRACT, `required_fix` MUST be derived from:

Decision Surface:

- "Define which outcomes are decision surfaces with system-specific names (stage, queue, entity type). Include at least one pre-validation gate (scatter/withhold)."

Authority Placement:

- "State where truth lives for deciding desired state. Declare authority directionality (normative/descriptive/transitional). Identify non-authoritative stores."

Failure Model:

- "Name failure classes with deterministic/transient distinction. State forbidden irreversible actions. Specify enforcement action per class (halt, skip-and-record, or bounded retry). 'Log and continue' is not enforcement."

Identity Semantics:

- "Define canonical identity for governed entities and any normalization/uniqueness rules."

Operational Bounds:

- "Declare at least one bounded execution constraint (concurrency limit, backpressure mechanism, pre-validation gate, or completion accounting). State what the system does when the bound is exceeded. Bounds must be system-enforced, not user-supplied."

Authoring Mode:

- "Declare at document top whether this contract is normative, descriptive, or transitional. This determines whether code-contract divergence is a code defect or a contract defect."

---

11. Extraction Protocol Guidance

When contracts are authored via automated extraction (SOTEP or equivalent), this guidance applies. Advisory, not normative for the lint.

1. **Extraction output is a draft.** It is never directly authoritative. A human must review every extracted claim against design intent before it becomes normative.

2. **Hack detection.** Extracted truths that appear to be workarounds (single-commit additions, TODO-adjacent code, unreachable paths, panic-driven constraints) MUST be flagged as `provisional` rather than promoted to invariant status.

3. **Liveness check.** Every code citation in an extracted contract must be verified to exist at extraction time. Functions, handlers, or stages referenced must pass an existence check (grep/import). Phantom references are extraction defects.

4. **Design intent filter.** If the author has provided a design intent document or architectural principles, extracted truths that conflict with stated intent MUST be flagged as divergences, not promoted as governing truths. The finding is: "code may need fixing," not "contract should match code."

5. **Attack review directionality.** When a normative contract is subjected to adversarial review:

    - The contract is the authority. Code divergence is a code defect.
    - A finding that proposes adding complexity to match current code behavior against stated design simplicity MUST be rejected.
    - The reviewer MUST check: "Does this finding violate the author's stated design principles?" If yes, reject.
    - "Intentionally unbounded" is a valid design choice. Attack reviews MUST NOT add constraints where the contract explicitly states no constraint exists.

---

12. Non-Goals

The OSC-K lint MUST NOT:

- Judge whether the contract's design choices are good.

- Enforce specific architectures or patterns.
- Require enumerating every DS exhaustively.
- Require machine-readable clause IDs.
- Replace Attack Types triage discipline.
- Penalize brevity. (Brevity is a defense against review-amplified drift.)
- Require justification for design choices. (A contract states rules, not arguments.)

---

13. Versioning and Immutability

This specification is immutable for version v0.3.

Any change beyond typos requires at least a 0.1 version bump.

Major semantic changes (new kernel elements, removed elements, changed evaluation logic) require a +1 bump.

---

14. Changelog from v0.2

Change	**Section**	**Motivation**
Added kernel element: Operational Bounds	§4.5	Production failures from unbounded execution, human-as-control-loop
Added kernel element: Authoring Mode Declaration	§4.6	SOTEP/attack-review authority inversion
Added Concrete Instantiation Rule	§6	Generic fill-ins passed v0.2 with zero operational value
Expanded Failure Model: deterministic/transient distinction	§4.3	Retry pyramids on deterministic failures wasted 11+ hours
Expanded Failure Model: enforcement action requirement	§4.3	Catch-and-log validators were decorative
Expanded Failure Model: banned "log and continue"	§4.3	Indistinguishable from no enforcement in production

Change	Section	Motivation
Expanded Decision Surface: scatter/withhold	§4.1	Bulk operations on unvalidated entity lists caused 97%+ failure rates
Added Authority Directionality to §4.2	§4.2	Attack reviews reversed correct architectural fixes
Added Document Structure Guidance	§7	OSC-K position flexibility was unstated; engineers found framework-ordered docs unnavigable
Added Extraction Protocol Guidance	§11	SOTEP promoted unauthorized code into governing contracts
Added ABSTRACT status	§9	Needed for Instantiation Test results
Evaluation: PASS requires instantiation	§8	v0.2 PASS was achievable with worthless abstract content
BLOCK includes "log and continue"	§8	Decorative enforcement must be caught, not just warned