

SLQCS v0.1

Standard for Logical Quantum Computational Systems

Substrate-Agnostic Definition, Conformance Profiles and Evidence Protocol

A Logical Quantum Computational System preserves, transforms, composes and resolves quantum-semantic information so that every supported continuation produces the declared quantum observables within a predefined error contract, demonstrated by reproducible tests, negative controls and independent verification.

Status: Candidate Technical Standard

Version: 0.1

Publication date: 1 October 2026

Document series: SLQCS Candidate Standard

SUBSTRATE-AGNOSTIC PRINCIPLES | VERIFIABLE EXECUTION | HONEST CLAIMS

SLQCS Candidate Technical Standard

Version 0.1 | 1 October 2026

SLQCS at a glance

The one-page operational map. The following sections provide the complete normative definitions, evidence rules and qualification protocol.

1. Purpose

Define logical quantum computation independently of physical, digital or hybrid substrate. Conformance follows preserved semantics and supported continuations - not material, storage format or marketing category.

Fundamental criterion

A system MUST preserve every distinction required to produce the correct observable consequences under every continuation promised by its declared execution profile.

2. Logical execution unit

Defined by semantic contract, not by storage format.

PREPARE	EVOLVE	COMPOSE	INTERFERE	READ

3. Supported continuation

Two states are equivalent only if every supported continuation produces the same declared observables within tolerance. A matching current histogram is insufficient when a later operation can distinguish the states.

4. Conformance vector

$D = (D_{\text{output}}, D_{\text{phase}}, D_{\text{relation}}, D_{\text{trajectory}}, D_{\text{conditional}})$. Every mandatory component must satisfy its own frozen tolerance; one strong dimension cannot compensate for another.

5. Oracle rule

An oracle is qualified by its declared action under controlled use, composition, phase kickback, inversion when supported and later continuations - never by agreement with a precomputed final answer alone.

6. Conformance profiles

CORE	substrate-agnostic semantics
GATE	discrete gate / logical-qubit
ORACLE	logical oracle and query action
DYNAMIC	conditional / deferred resolution
HAMILTONIAN	scientific / operator execution
PHOTONIC	bosonic / photonic
PHYSICAL-QEC	physical fault tolerance

7. Scale qualification

Report width, instruction set, depth, connectivity, control/readout structure, verifier and resources. Width alone is insufficient.

8. Complexity honesty

Report separately: (1) semantic correctness, (2) executed scale, and (3) computational efficiency. None implies the others.

9. Evidence package

Bind input -> implementation -> execution -> result -> verification, with negative controls, failures, hashes and qualification records.

10. Qualification campaign

A Semantic core | B Independent relations | C Scale ladder | D Specialized profile | E Blind external run

Mandatory core conformance - C1 to C12

C1	State sufficiency	C7	Conditional evolution
C2	Relative-phase causality	C8	Reversibility / echo
C3	Joint relational state	C9	Negative controls
C4	Deferred resolution	C10	Numeric contract
C5	Lawful transformation	C11	Scaling preservation
C6	Measurement semantics	C12	Independent verification

North-star definition

A Logical Quantum Computational System preserves, transforms, composes and resolves quantum-semantic information so that every supported continuation produces the declared quantum observables within a predefined error contract, demonstrated by reproducible tests, negative controls and independent verification.

1. Status and use of this document

SLQCS v0.1 is a candidate technical framework. It is intended to support technical qualification, procurement diligence, independent evaluation and precise public reporting. It is not an ISO, IEC, IEEE or other accredited consensus standard, and it does not replace sector-specific safety, security or regulatory requirements.

The words MUST, MUST NOT, SHOULD, SHOULD NOT and MAY indicate requirement strength. A system may claim SLQCS conformance only for an explicitly declared execution profile, instruction set, workload domain, width, depth, connectivity, readout model and numeric contract.

Conformance is never inferred from substrate, branding, qubit count, a single final answer, or resemblance to an existing physical machine.

2. Purpose

SLQCS defines when a system may be described as implementing logical quantum computation independently of whether its substrate is superconducting, photonic, trapped-ion, neutral-atom, spin, topological, analog, digital or hybrid.

The standard separates five questions that are often conflated:

1. Does the implementation preserve the semantic distinctions required by its declared quantum operations?
2. Does it produce the declared observable consequences under all supported continuations?
3. At what width, depth and structural complexity has this been demonstrated?
4. What time, memory, energy and cost are required?
5. Has the result been independently reproduced or verified?

Correctness, scale, efficiency, physical implementation and quantum advantage are separate claims. None implies the others.

3. Scope

SLQCS applies to systems that claim to execute a declared class of logical quantum computations. It covers:

- gate-model execution;
- oracle and deferred-resolution execution;
- Hamiltonian and scientific-computation profiles;
- photonic or bosonic profiles;
- physical fault-tolerant implementations;
- digital or hybrid implementations that preserve the same declared logical semantics.

SLQCS does not certify:

- universal quantum computation unless the declared profile and evidence support that claim;

- computational advantage over the best classical method;
- physical nonclassicality, entanglement of matter or photons, or use of a QPU;
- security, safety, cryptographic suitability or production readiness;
- correctness outside the declared instruction set and workload domain.

4. Normative definitions

4.1 Logical Quantum Computational System (LQCS)

A computational system whose internal representation and transition rules preserve the information required to implement its declared quantum operations and their observable consequences.

4.2 Logical Quantum Computational Unit (LQCU)

A computational object that supports a declared set of quantum-semantic operations and preserves the information required by supported future continuations. An LQCU is defined by its semantic contract, not by its storage format or physical material.

4.3 Supported continuation

Any future operation sequence that the declared execution profile promises to support. State equivalence is judged against all supported continuations, not only against the current measurement.

4.4 Observable consequence

A declared result that can be checked at the interface of the system, including a bit string, probability, expectation value, correlation, eigenphase, energy, witness, return probability or another profile-specific observable.

4.5 Declared execution profile

The complete boundary of the conformance claim: instruction set, input class, width, depth, connectivity, measurement model, numeric representation, tolerance, supported control flow and excluded operations.

4.6 Deterministic execution

Execution in which the same accepted input, configuration and version produce the same declared result within the numeric contract. For an indeterminate observable, that result MAY be an exact distribution, expectation value, reproducible logical resolution or another explicitly declared object. It MUST NOT be misrepresented as a physical hardware shot. Determinism does not by itself prove quantum-semantic correctness.

4.7 Simulator

A system whose computational contract is to model or emulate a separately specified target machine, physical device, state representation or dynamical model. Classification **MUST** be operational, not rhetorical: changing the label does not change the evidence obligation. Internal representation and substrate alone do not determine whether a system is a simulator. A direct logical implementation **MAY** use a physical, digital or hybrid substrate if it independently satisfies the declared semantic contract under supported continuations.

4.8 Direct logical implementation

An implementation whose declared contract is to perform the logical operations themselves rather than reproduce the behavior of a separate target machine. A direct implementation **MUST** satisfy the applicable C1-C12 requirements. Directness does not imply physical nonclassicality, universal capability, computational advantage or efficiency.

4.9 Logical quantum oracle

A callable logical operation that implements a declared mapping or transformation inside a quantum computation. An oracle **MAY** be physical, digital or hybrid, and **MAY** be primitive or composite. Its conformance is determined by its operational action under every supported query and continuation, including controlled application, composition, interference or phase kickback, inversion when declared, and measurement consequences.

An oracle is not established by returning a known benchmark answer. A precomputed answer, lookup shortcut or hidden use of the expected result that bypasses the declared transformation is nonconforming.

4.10 Physical, digital and hybrid implementation descriptors

These terms describe substrate, not correctness:

- A physical implementation realizes the declared semantics through controlled physical degrees of freedom and **MUST** report its preparation, noise, measurement and error-control model.
- A digital implementation realizes the declared semantics through digital state and transition mechanisms and **MUST** report its logical interface, numeric contract and resource boundary. It is not automatically a simulator merely because its substrate is general-purpose digital hardware.
- A hybrid implementation divides the declared computation across physical and digital components and **MUST** identify the semantic boundary between them.

None of these descriptors grants conformance without the applicable evidence.

4.11 Evidence package

A versioned, inspectable collection binding input, implementation, execution, result and verification. It includes positive tests, negative controls, environment data, logs, tolerances, hashes and qualification statements.

4.12 Qualification campaign

A frozen collection of tests and evidence procedures used to determine conformance for a declared profile and scale.

5. Fundamental criterion

A logical quantum computational implementation **MUST** preserve every distinction required to produce the correct observable consequences under every continuation that its declared execution profile promises to support.

Two states are equivalent for a declared profile only if no supported continuation can distinguish them through a declared observable:

$$S1 ==F S2 \text{ iff } P(y \mid C, S1) = P(y \mid C, S2)$$

for every supported continuation C and declared observable y , within the frozen numeric tolerance.

Therefore, matching a single output or histogram is insufficient when a later supported operation could distinguish the underlying states.

6. Logical execution model

An SLQCS implementation exposes a semantic chain:

1. Prepare - create or accept a declared logical input.
2. Evolve - apply declared transformations.
3. Compose - combine logical units and relational structures.
4. Interfere - preserve and use phase-dependent alternatives where required.
5. Read - resolve the declared observable under the measurement contract.

The implementation **MAY** use any internal representation. Conformance depends on observable equivalence under supported continuations, not on whether a state vector, tensor network, amplitudes, samples, symbolic relations, geometry, analog dynamics or another representation is used.

For oracle execution, the semantic chain **MUST** include the oracle's declared action, not only its final answer. Where applicable, qualification **MUST** test controlled oracle use, phase kickback, composition with later operations, repeated queries, inverse behavior and evaluator-selected inputs. The same oracle prefix **SHOULD** remain valid under more than one supported continuation.

7. Core conformance requirements

C1. State sufficiency

Requirement. At every execution point, the implementation **MUST** retain all information required by every supported future continuation. It **MUST NOT** collapse distinct logical states into one internal state when a supported later operation can distinguish them.

Required evidence. The evidence package **MUST** include paired prefixes that appear equivalent under immediate readout but become distinguishable after a frozen continuation. The implementation **MUST** produce the correct later observables for both.

Minimum tests. Basis-state preparation; superposition preparation; paired-state continuation tests; prefix reuse across at least two different continuations; serialization or checkpoint restoration if supported.

Failure conditions. Output lookup; hidden use of expected answers; premature measurement; loss of phase or relational information required later; a representation that succeeds only when the continuation is known in advance.

C2. Relative-phase causality

Requirement. Relative phase **MUST** remain causally available whenever a supported later operation converts it into an observable difference.

Required evidence. Frozen phase-sensitive circuit pairs **MUST** differ only in a phase operation and **MUST** produce the predicted different observables after the same continuation.

Minimum tests. H-Z-H or equivalent phase kickback; H-T-H; Bell states with relative phase; phase sweeps; inverse phase recovery; at least one non-Clifford phase test where supported.

Failure conditions. Treating states with different operationally relevant phase as equivalent; reproducing only computational-basis outputs; phase results dependent on disclosed expected answers.

C3. Joint relational state

Requirement. The system **MUST** preserve joint information that cannot be reduced to independent local summaries whenever the declared workload requires correlated or entangled relations.

Required evidence. Tests **MUST** include multi-unit correlations under multiple measurement bases or continuations. Local marginals alone are insufficient.

Minimum tests. Bell correlations; GHZ parity; CHSH or an equivalent correlation witness; Peres-Mermin or another contextuality-style relation; topology-specific multi-unit tests.

Failure conditions. Correct local outputs with incorrect joint distribution; factorization of a required relational state; correlation generated only at final readout without supporting intermediate continuations.

C4. Deferred resolution

Requirement. Alternatives that remain unresolved under the declared computation **MUST** stay operationally available until the declared resolution or measurement step.

Required evidence. The evaluator **MUST** be able to insert or select later continuations that reveal different consequences of the unresolved structure.

Minimum tests. Iterative phase extraction; oracle branches; delayed-basis choice; continuation selected after prefix execution; at least one multi-step deferred-resolution workload.

Failure conditions. Selecting a branch before the declared measurement; returning a deterministic answer without preserving the information needed for alternative continuations; inability to reuse the same prefix under a different supported readout.

C5. Lawful transformation

Requirement. Every accepted operation **MUST** implement its declared logical quantum transformation within the numeric contract and in the correct order.

Required evidence. Gate-level or operation-level reference tests, composition tests, inverse tests and cross-family workloads **MUST** be supplied for the declared instruction set.

Minimum tests. Single-unit gates; controlled operations; non-commuting operation order; at least one non-Clifford operation where claimed; multi-step algorithms with evaluator-supplied parameters.

Failure conditions. Unsupported operation silently approximated or ignored; commutation applied where invalid; operation semantics dependent on a known benchmark answer; incorrect behavior under composition.

C6. Measurement and readout semantics

Requirement. Readout **MUST** follow the declared measurement contract. Determinate inputs, indeterminate inputs, distributions, expectation values and deterministic observables **MUST** be distinguished and reported correctly.

Required evidence. The system **MUST** declare whether it returns samples, exact probabilities, deterministic logical values, expectation values or another observable. Tests **MUST** compare the returned object with the correct reference for that declared object.

Minimum tests. Basis readout; rotated-basis readout; Born-rule or probability tests where distributions are claimed; deterministic observable tests; indeterminate initialization; multi-unit joint readout.

Failure conditions. Presenting an expectation value as a sample; presenting one deterministic logical result as a physical-shot distribution; suppressing unresolved multiplicity; undeclared post-selection.

C7. Conditional state evolution

Requirement. When the declared profile supports dynamic control, later operations **MUST** respond correctly to earlier measurements, logical predicates or resolved information.

Required evidence. Tests **MUST** include both branches of each conditional path and **MUST** freeze the condition before execution. Mid-circuit measurement, feed-forward or logical equivalent behavior **MUST** be declared precisely.

Minimum tests. Conditional gate application; iterative phase estimation; branch inversion; nested conditions where supported; replay with forced branch outcomes; condition-dependent negative controls.

Failure conditions. Only one branch tested; branch result precomputed from the benchmark; condition evaluated from the expected answer instead of the executed state; incorrect branch composition.

C8. Reversibility and echo

Requirement. For operations declared reversible, execution followed by the declared inverse **MUST** restore the prior logical state within the numeric contract.

Required evidence. Forward-inverse and forward-perturbation-inverse campaigns **MUST** record intermediate checks, return probability or state witnesses, and maximum observed deviation.

Minimum tests. Random reversible circuits; non-commuting sequences; deliberate perturbation; depth sweep; width sweep; checkpointed intermediate observables.

Failure conditions. Final answer restored while intermediate semantics are inconsistent; tolerance selected after results are known; inverse implemented as reset; perturbation fails to change the predicted return result.

C9. Negative controls and fault injection

Requirement. A conformance package **MUST** demonstrate not only expected success but expected failure. Deliberate semantic violations **MUST** produce the predicted change in observables or verification outcome.

Required evidence. The evaluator **MUST** freeze fault types and expected signatures before execution. The package **MUST** include omitted gates, changed phases, swapped controls, corrupted relations, altered conditions and tolerance violations where applicable.

Minimum tests. At least one negative control for phase, relation, transformation, measurement and conditional behavior; hash mismatch detection; implementation/version mismatch detection.

Failure conditions. Negative controls pass unchanged; verifier accepts corrupted evidence; expected failure signatures are not specified before execution; only positive benchmark cases are reported.

C10. Numeric contract

Requirement. Precision, rounding, approximation, overflow behavior and comparison tolerances **MUST** be declared before qualification. Every reported result **MUST** identify the applicable contract.

Required evidence. Absolute and relative error metrics, precision sweeps, stability tests, worst-case observed deviation and pass thresholds **MUST** be included.

Minimum tests. Small exactly solvable cases; phase values near boundaries; deep cancellation; accumulation stress; repeated deterministic runs; cross-platform execution if portability is claimed.

Failure conditions. Tolerance chosen after observing results; exactness claimed without a defined numeric domain; hidden fallback to higher precision; unstable results under identical inputs.

C11. Scaling preservation

Requirement. Correctness demonstrated at small scale **MUST** be re-established across the declared width, depth and structural-complexity range. Qubit or unit count alone is insufficient.

Required evidence. A scale ladder **MUST** report width *W*, instruction set *I*, depth *D*, connectivity, control structure, measurement model, runtime, memory, energy when available and verification method.

Minimum tests. At least four increasing scale points; depth and width varied separately; non-trivial relational structure; at least one evaluator-supplied instance at the largest independently verified scale.

Failure conditions. Roadmap figures presented as executed evidence; width increased while depth or structure collapses; only trivially classically reducible instances used without disclosure; resource figures omitted.

C12. Independent verification

Requirement. A party independent of the implementation team **MUST** be able to supply inputs, execute or observe execution, obtain raw results and verify the declared observables.

Required evidence. The independent evaluator's identity, role, access level, selected workloads, verification method and limitations **MUST** be recorded. Company observation is not independent reproduction.

Minimum tests. Evaluator-supplied circuits or problem instances; blind expected values where practical; signed inputs and outputs; reproducible verification code; disclosure of assistance or materials supplied by the vendor.

Failure conditions. Vendor-selected outputs only; evaluator cannot inspect inputs or raw outputs; compensation or conflicts not disclosed; positive commentary treated as technical verification.

8. Conformance vector and decision rule

Conformance is evaluated as a vector, not as one aggregate score:

$$D(A,B;x) = (D_output, D_phase, D_relation, D_trajectory, D_conditional)$$

For every mandatory dimension i , the measured deviation **MUST** satisfy the frozen threshold:

$$D_i \leq \epsilon_i$$

A strong result in one dimension **MUST NOT** compensate for failure in another mandatory dimension. Every applicable C1-C12 criterion is reported as PASS, FAIL, NOT APPLICABLE or NOT TESTED, with evidence references. NOT APPLICABLE requires a profile-based justification. NOT TESTED is not conformance.

8.1 Complexity honesty rule

Three properties **MUST** be reported independently:

1. Semantic correctness - whether the system produces the declared quantum observables under the qualified profile.
2. Executed scale - the width, depth, structure and problem class actually run.
3. Computational efficiency - the time, space, energy and cost used relative to the problem class and baseline.

None implies the others. Correct execution does not by itself prove advantage; large width does not prove semantic difficulty; fast execution does not prove that the workload was general or classically hard.

9. Conformance profiles

9.1 SLQCS-CORE

Mandatory substrate-agnostic logical semantics. C1-C6, C8-C12 are mandatory. C7 is mandatory when dynamic control or conditional continuation is claimed.

9.2 SLQCS-GATE

Discrete gate and logical-qubit profile. The report **MUST** identify gate set, decomposition policy, control semantics, connectivity model and measurement bases.

9.3 SLQCS-ORACLE

Logical-oracle profile. The report **MUST** identify the oracle mapping, input domain, query interface, controlled action, phase behavior, inverse support, composition rules and prohibited shortcuts. C1, C2, C4, C5 and C9 receive expanded testing. Returning a precomputed answer is not oracle

execution.

9.4 SLQCS-DYNAMIC

Dynamic-control and deferred-resolution profile. C4 and C7 receive expanded testing, including late-bound continuation, conditional execution, mid-circuit information use where supported, and evaluator-selected readout.

9.5 SLQCS-HAMILTONIAN

Hamiltonian and scientific-computation profile. The report MUST identify operator construction, term set, coefficients, commutation structure, ansatz or state preparation, observable extraction and numeric reference.

9.6 SLQCS-PHOTONIC

Bosonic or photonic profile. The report MUST identify encoding, mode structure, loss model, detector/readout semantics, post-selection and any physical nonclassicality claim.

9.7 SLQCS-PHYSICAL-QEC

Physical fault-tolerant profile. The report MUST identify physical and logical error rates, code family, decoder, syndrome cycle, logical operations, overhead and evidence that error correction suppresses logical error as scale increases.

Passing SLQCS-CORE does not authorize a physical-QPU, nonclassicality, fault-tolerance or quantum-advantage claim unless the relevant specialized profile is also satisfied.

10. Qualification campaign

An SLQCS qualification campaign has five stages:

Stage A - Semantic core

Freeze profile, instruction set, numeric contract and reference cases. Qualify C1-C6 at small scale with exactly solvable tests.

Stage B - Independent relations

Qualify multi-unit relations, conditional evolution, reversibility and negative controls. Emphasize C3, C7, C8 and C9.

Stage C - Scale ladder

Repeat the applicable core tests across increasing width, depth and structural complexity. Separate width scaling from depth scaling. Record time, memory, energy and cost without implying advantage.

Stage D - Specialized profile

Run the profile-specific suite for gate, oracle, dynamic, Hamiltonian, photonic or physical-QEC claims.

Stage E - Blind external run

An independent evaluator supplies at least one previously undisclosed valid input and at least one negative control. The input, environment, execution and result are cryptographically bound and externally verified.

11. Scale qualification

Every scale claim MUST report:

- W: logical width;
- I: instruction set and operation mix;
- D: logical depth and critical path;
- connectivity or relational structure;
- measurement and control model;
- instance generation method;
- verifier and reference method;
- wall-clock time, engine time and queue time separately;
- peak memory and hardware configuration;
- energy or power measurement method when energy is claimed;
- repetitions, variance and deterministic replay behavior.

Width alone is insufficient. A larger width with trivial structure does not supersede a smaller but deeper or more relationally demanding execution.

12. Evidence package

The evidence package MUST bind the complete chain:

input -> implementation -> execution -> result -> verification

Recommended package structure:

```
standard/  
  profile.yaml  
  numeric-contract.yaml  
input/
```

```
circuits-or-problems/  
  generators/  
  hashes.txt  
implementation/  
  version.txt  
  configuration.yaml  
run/  
  environment.yaml  
  command.txt  
  logs/  
result/  
  raw/  
  normalized/  
verification/  
  reference/  
  scripts/  
  report.json  
negative-controls/  
  cases/  
  expected-signatures.yaml  
failures/  
  observed/  
qualification/  
  conformance-report.pdf  
  evaluator-statement.pdf
```

Every artifact SHOULD be content-addressed. The package MUST declare any unavailable proprietary component and MUST provide enough interface evidence for an evaluator to verify the claim without access to the protected implementation.

13. Verification protocol

1. Freeze the profile and all pass criteria.
2. Hash inputs, generators, configuration and verification code.
3. Separate test generation from execution where practical.
4. Include positive cases, boundary cases and negative controls.
5. Capture raw outputs before normalization or interpretation.
6. Verify results using an independent method.
7. Record failures, exclusions and reruns.
8. Repeat deterministic cases to confirm reproducibility.
9. Run the scale ladder without changing the semantic contract.
10. Issue a signed conformance report with explicit limitations.

14. Claims and reporting rules

14.1 Semantic correctness

May be claimed only for the declared profile and tested domain. Use language such as: "Conforms to SLQCS-CORE v0.1 for the declared instruction set and qualification range."

14.2 Executed scale

Must distinguish company-executed, independently executed, independently observed and roadmap scale. Future targets **MUST NOT** be presented as completed qualification.

14.3 Computational efficiency

Runtime, memory, energy and cost require a defined workload, hardware configuration, baseline, measurement boundary and quality target. An internal benchmark **MUST NOT** be generalized beyond its tested conditions.

14.4 Physical claims

SLQCS conformance does not prove physical nonclassicality, physical qubits, entangled particles or operation of a QPU. Those claims require separate physical evidence.

14.5 Quantum advantage

Logical correctness, large width or fast execution does not by itself prove quantum advantage. Advantage requires comparison with the best relevant classical method, including preprocessing, approximation, hardware, energy, verification and total cost.

14.6 Direct-execution and non-simulation claims

A system **MAY** claim direct logical quantum execution rather than simulation of a target QPU only when it declares no separate target machine as the object being emulated and demonstrates the applicable semantic obligations through the SLQCS evidence chain. The claim **MUST** be limited to the qualified instruction set, workload domain and scale.

Such a claim does not, by itself, assert physical qubits, physical nonclassicality, a QPU, universal quantum computation or quantum advantage.

14.7 Oracle claims

An oracle claim **MUST** identify the declared mapping, input domain, supported query semantics, controlled action, phase behavior, composition, inversion when supported, observable contract and verification method. An oracle claim **MUST NOT** be based solely on agreement with a final answer. Evidence **MUST** distinguish execution of the declared transformation from lookup, precomputation or benchmark-specific shortcuts.

15. Conformance statement template

A public conformance statement **SHOULD** contain:

- system and version;
- SLQCS version and profile;

- declared workload domain;
- supported and excluded operations;
- qualified width, depth and structure;
- applicable C1-C12 results;
- numeric contract;
- evidence package identifier and hash;
- independent evaluator and access level;
- resource measurements;
- limitations and unresolved items;
- date and signature.

Appendix A - Minimum reference test catalogue

Requirement	Example positive evidence	Example negative control
C1	Same prefix, two later continuations	Merge distinguishable prefixes
C2	H-T-H, Bell phase, phase sweep, oracle phase kickback	Remove or invert phase
C3	Bell, GHZ, CHSH, Peres-Mermin	Break one relation or control
C4	IPE, late-bound readout	Resolve branch early
C5	Gate/oracle action, composition and inverse	Swap non-commuting operations or bypass oracle action
C6	Basis and rotated-basis readout	Mislabel sample as expectation
C7	Both conditional branches	Force wrong branch
C8	Forward-perturbation-inverse	Replace inverse with reset
C9	Frozen fault signatures	Corrupt evidence hash
C10	Precision and boundary sweep	Exceed frozen tolerance
C11	Width/depth ladder	Increase width while removing structure
C12	Evaluator-supplied blind input	Vendor-only selected case

Appendix B - Machine-readable result fields

Each case SHOULD report: case_id, profile, criterion, input_hash, implementation_version, environment_hash, start_time, end_time, raw_result_hash, observable_type, expected_value, observed_value, metric, tolerance, verdict, verifier, independence_level, notes.

Appendix C - Change control

Changes to the standard MUST preserve versioned definitions, test identifiers and evidence schemas. A qualification report MUST identify the exact SLQCS version used. Breaking changes require a new minor or major version and MUST NOT retroactively alter a prior verdict.

Closing principle

SLQCS evaluates what a system preserves and executes, not what material it is built from. The burden of proof is operational: frozen semantics, distinguishing continuations, declared tolerances, negative controls, scale evidence and independent verification.

Substrate-agnostic principles. Verifiable execution. Honest claims.