Supplement S1: Mechanism disclosures and claim register

to Securing the Semantic Supply Chain: Meaning as a Foundation of Trust Infrastructure

Supplement to the research article

Dr. Carsten Stöcker
Spherity GmbH
carsten.stoecker@spherity.com
ORCID 0009-0003-7417-8872

Version 1.0, 16 November 2026 doi:10.5281/zenodo.24293613



Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). © 2026 Dr. Carsten Stöcker.

Contents

S1.1 Produce **2**

S1.2 Release **4**

S1.3 Distribute **5**

S1.4 Verify **7**

S1.5 Decide **10**

S1.6 Monitor **15**

S1.7 Claim register **19**

Numbers in square brackets refer to the reference list of the article.

This supplement belongs to the article *Securing the Semantic Supply Chain: Meaning as a Foundation of Trust Infrastructure* (C. Stöcker, 2026).

Each mechanism is described so that a skilled engineer can build it. Variants are listed so that foreseeable alternative embodiments are disclosed too. "Prior art" names the closest work found; "Delta" is what the article adds. The harness implements tested software paths for M15, M17, M18 and M19, protocol simulations for M24, and a synthetic M31 example; the files and limits are identified below. This is not a claim that every production obligation of those mechanisms is implemented.

S1.1 Produce

M1 — Semantic build pipeline.

1. *Source.* Semantic artefacts are kept in version control with signed commits, protected branches and two-person review. Every change of meaning references a rulebook change record and its rationale.
2. *Build.* A hermetic, isolated, ephemeral builder (SLSA Build Level 3) generates every published form from one source model: JSON-LD contexts, SHACL bundles, SD-JWT VC Type Metadata, CBOR-LD type tables, catalogue entries, compiled contexts (M5).
3. *Verify.* The pipeline runs, and records as in-toto attestations: SHACL meta-validation; negative test vectors per attack class; cross-serialisation equivalence (M4); processing bounds (step 5); the semantic diff (M2).
4. *Release, publish, distribute.* Signing under M10/M11; logging under M9; distribution under M8.
5. *Processing bounds as a release gate.* For each artefact the pipeline measures, on production and adversarial corpora: maximum expansion size; RDFC operation count; SHACL validation cost; OWL import-closure size; `extends` chain depth. It publishes declared bounds with the artefact. An artefact whose own corpus exceeds its bounds is not released. Verifiers enforce the declared bounds (M19).

Prior art: SLSA 1.2; in-toto; Sigstore. *Delta:* semantic gates (diff, equivalence, bounds) in place of unit tests.

M2 — Expansion-level semantic diff that enforces versioning. The publisher's pipeline expands a regression corpus of positive and negative instances under the previous and the candidate release, canonicalises both with RDFC-1.0 under bounds, and classifies the difference: *identical* (patch), *additive only* (minor), or *any changed or removed triple or changed SHACL outcome* (major). A major change cannot be released under an existing identifier; it needs a new context URL or `vct`, a new rulebook version and a deprecation notice. The result is recorded in a signed `semantic-diff/v1` attestation. The classification is relative to the corpus: changes the corpus does not exercise are not detected, so the pipeline also applies conservative syntactic rules (any change to an existing term definition is at least minor). Additional detail:

- The regression corpus holds, per type and term, at least one positive and one negative instance, plus every instance from the published test vectors.
- Classification uses the RDFC-canonical N-Quads of each corpus instance under both releases, plus the SHACL validation report per instance. *Patch:* all identical. *Minor:* new quads appear only for new terms, no existing quad changes, all SHACL outcomes unchanged. *Major:* anything else.

- The `semantic-diff/v1` attestation contains: previous and candidate release digests, corpus digest, class, and the list of changed instances.
- *Variants:* diff over OWL axioms in addition to instances; diff over SD-JWT VC claim mappings through Type Metadata; a "breaking-query" rule that classifies as major any change that alters the result of a registered SPARQL query.

M3 — Closure manifests. For each release, a canonical JSON manifest identifies the processing-aware context dependency closure, including loader metadata Λ and processor semantics P; the content digests of the vocabulary closure V, including OWL imports, SKOS schemes, SHACL graphs, extends parents, rulebooks, crosswalks and status profiles; and the identity and configuration of the interpretation engine E. The semantic-closure digest uses the typed, domain-separated construction in S3.1.3 over the context-manifest digest, V's content digests and E. SHACL severity changes (`sh:Violation` → `sh:Warning`) and openness changes (`sh:closed true` → `false`) are classified as major by M2. *Variants:* a Merkle tree over the members, so that verifiers can prove membership of one member; one manifest per credential type.

M4 — Cross-serialisation equivalence attestation. For every corpus instance, the pipeline expands the instance through every published path (JSON-LD; CBOR-LD encode → decode; YAML-LD → JSON-LD; SD-JWT VC claims mapped through Type Metadata where a mapping is declared), canonicalises each result with RDFC-1.0 under bounds, and compares the digests. It emits a `semantic-equivalence/v1` attestation listing artefact digests, corpus digest and result; a release requires *equal* for every instance. The canonical RDF digest compared here establishes L2 equivalence for the tested corpus instances. It is distinct from M22's semantic-closure digest, which commits to the context-manifest digest, vocabulary-closure content digests and interpretation engine identity and configuration (S3.1.3).

M5 — Signed compiled contexts. The pipeline compiles each released context into an **active-context snapshot**: the processed term-definition table, with `@protected` flags, IRI mappings, type coercions and container rules (including which terms create lists, sets or blank nodes). The snapshot is serialised deterministically (for example, as canonical CBOR), signed under M8, logged under M9, and linked to its source context by an M4 attestation over the corpus. For a restricted, declared JSON-LD language, verifiers load snapshots instead of contexts, so expansion becomes table lookup, and admission (M18) knows exactly which constructs can create blank nodes. In general an active context depends on processing state, base IRI, propagation and scope, so the compiler and its supported language must themselves be certified; a corpus attestation (M4) is evidence, not a proof of equivalence. *Variants:* a snapshot that also contains the CBOR-LD term-codec map; pre-expanded frames for frequent credential types; a JSON-LD processor API that accepts a signed snapshot.

M6 — YAML-LD as source only. YAML-LD is accepted only at the source stage, parsed with a YAML 1.2.2 core-schema parser in a sandboxed process; local tags other than those YAML-LD defines are rejected, alias expansion is counted and bounded, only one document is allowed, duplicate keys are rejected. Only the built JSON-LD and its expanded-RDF digest are published; a CI rule rejects YAML-LD that does not round-trip to the same RDFC digest.

M7 — Machine-generated semantics as untrusted source. Changes drafted by LLMs or semantic compilers enter only as source proposals, labelled with the generator's identity, version and input digest. They pass the M1 gates plus a second human reviewer for any major diff. Provenance records the machine involvement. A generator never holds a signing share.

M28 — Governed crosswalks. A crosswalk (for example traceability vocabulary ↔ GS1 ↔ schema.org ↔ HS; `vct` ↔ JSON-LD types) is released as a semantic artefact: a mapping set (for example SSSOM) with a set-level closure digest, the digests of both vocabularies it maps,

a test corpus whose mapped instances must expand to the expected target quads, an M2 diff between releases, signing and logging. Decisions that use a mapping record its digest (M22).

S1.2 Release

M10 — Provenance-gated signing inside the HSM or TEE. The publisher's key signs only if the signing boundary itself has verified (a) build provenance from an allowed builder, (b) every required test attestation, (c) a semantic-diff class consistent with the requested version, and (d) for timestamp and snapshot roles, that the target metadata is already logged. The check runs inside the HSM firmware extension or the attested TEE, not in the build system that asks for the signature. Variants: vendor HSM policy engine; a confidential-computing enclave (AMD SEV-SNP, Intel TDX) that releases its key share only after the check; MPC signing where every party evaluates the policy. Request format: the signing service receives the artefact digest, the requested role and version, and a bundle of attestations (SLSA provenance, in-toto verification attestations, `semantic-diff/v1`, and for timestamp/snapshot roles a transparency receipt). Inside the boundary it verifies each attestation's signature against pinned builder identities, checks that the diff class matches the version increment (patch/minor/major), and only then signs. The boundary logs every request and decision. This is the publisher-side form of a general pattern, **policy-gated key use**, which also applies to agents (companion paper in preparation).

M11 — Rebuild-gated threshold release. Two or more independent builders (different organisations and infrastructure) each hold one share of a threshold signing key (for example FROST, RFC 9591, or threshold ECDSA). A builder contributes its share only if its own build from the same source commit reproduces the candidate's artefact digests and attestations. A valid threshold signature therefore shows that t shares participated; with at most f corrupt or colocated participants, at least $t - f$ honest builders reproduced the artefact. Builders must be independently administered failure domains, and each share is bound to the exact release digest and build evidence. *Variants:* builders in attested TEEs; 2-of-3 for the timestamp role and 3-of-5 for root; reproducibility checked on expanded-RDF digests where byte-level reproducibility is impractical.

M12 — Role-specific post-quantum algorithms. Root: stateful hash-based signatures (LMS/HSS, RFC 8554; XMSS, RFC 8391; NIST SP 800-208), hybrid with ECDSA during transition, in an offline HSM with state management. Namespace delegations and targets: ML-DSA (FIPS 204; RFC 9964) hybrid with ES256, or SLH-DSA for archival copies. Snapshot and timestamp: ML-DSA or ES256 until the migration date, short expiry. Hybrid rules follow PoS §8.6: every component the policy marks mandatory must be present and verify, with no fallback to a classical-only proof. A post-quantum countersignature held outside a carrier is mandatory evidence, so it must be archived and renewed before the classical algorithm becomes unsafe (DEC-3).

M13 — Attested entropy provenance. Issuers, status-list producers and publishers declare the random-number-generator class they use in signed metadata (for example an SP 800-90B-validated source with an SP 800-90A DRBG, or a BSI AIS 20/31 functionality class from the 2024 reference v3.0: DRG.2, DRG.3, DRG.4, DRT.1, PTG.2, PTG.3, NTG.1); the declaration is backed by a module certificate or TEE evidence. Attestation cannot demonstrate the entropy of an observed output. Status-list indexes and cover positions are assigned by a **keyed pseudo-random permutation** over the list capacity, with the key held in the HSM. *Incomplete:* a published commitment to the key alone does **not** let an auditor verify the assignment, and does not exclude bias in which counter is assigned to which holder. A private audit needs a committed issuance sequence, verifiable evaluations and a defined audit relation, or a trusted

confidential auditor; this protocol is future work (PRV-2). A QRNG is one admissible source, not a requirement.

S1.3 Distribute

M8 — Update roles for meaning. Metadata, adapted from TUF:

- *root.json:* keys and thresholds for all roles; rotated by a threshold of old and new root keys.
- *targets.json* and per-namespace *delegations:* for each artefact, its path or URL, length, digest, version, and **status** (`current`, `deprecated`, `withdrawn`, with reason and an advisory reference, M30).
- *snapshot.json:* versions of all targets metadata (prevents mix-and-match).
- *timestamp.json:* digest of the snapshot, short expiry (minutes to hours).

Verifier rules: fetch and check in the order root → timestamp → snapshot → targets; require **every** object in the chain to be unexpired and linked by checked digests and versions, and fail closed (or return *unknown*) otherwise; keep durable **high-water marks for the metadata** (snapshot and timestamp versions) and **for each artefact's version**, and reject older ones unless a current, scoped downgrade signed by the targets role exists; record every accepted withdrawal and **never use a withdrawn version again for current authorisation**, whatever older metadata is later presented (historical decoding for replay remains allowed); obtain time from a trusted source with a declared maximum error ε (RFC 3161 response or Roughtime). Withdrawal status is signed by the targets role (threshold, offline). The profile names the **status-bearing object** and its maximum lifetime Δ_status: a fresh timestamp signed by a compromised online key can point to an older status object, so the timestamp's lifetime does not bound freshness. *Formal model (T5, T6; MC-1, MC-5):* a withdrawal takes effect within Δ_status + ε; a counter on artefact versions alone lets a replayed older "current" metadata object undo a withdrawal. *Variants:* the same roles carried in a registry API (for example EBSI-style registries, DID-linked resources); signed catalogue responses with status fields.

M9 — Transparency for semantic artefacts. Each release is registered as a signed statement (issuer, artefact type, identifier, version, digest, status, closure digest) in a transparency service following RFC 9943; the receipt is distributed with the artefact. Verifiers accept a release only with an inclusion proof under a checkpoint that is consistent with the checkpoints they have retained and cosigned by a quorum of a common, stateful witness set whose quorums intersect in an honest witness ($2t - n > f$); this excludes forks (T7). A monitor reads every leaf under accepted checkpoints and checks a declared conflict relation over (publisher, namespace, identifier, version, epoch, status); legitimate succession is not a conflict. Witness-set changes follow a separately specified reconfiguration rule. *Regulated variant:* the log is operated as an eIDAS qualified electronic ledger (Arts. 45k–45l; reference standards and specifications in CIR (EU) 2025/2531). *Other variants:* Sigstore Rekor; a permissioned ledger operated by a standards body.

M14 — Edge director with rollback counters. A per-ecosystem or per-site **director** signs, for each device class, the exact semantic bundle the devices must run and a **maximum offline window** W. The device stores the highest accepted bundle version in a TPM NV monotonic counter or secure-element storage and rejects lower versions; protection across a factory reset or TPM clear requires a defined device identity and re-enrolment procedure. If the last director timestamp is older than W, the device enters a declared degraded mode (escalate; accept only low-risk actions). Devices report their loaded bundle digest when connected (M23).

M15 — Codec binding outside the compressed payload, with a lifetime archive. *Experimental codec-binding implementation:* `src/cborld-binding.mjs`. *Lifetime retention, transparency integration and the variants below are deployment requirements, not implemented services.*

- *Manifest (decoding manifest, DST-12):* the registry-entry identifier; the decoder and processing-model identifiers and versions; codec semantics (type and URL codecs, typed-value codecs) and their options; every type table as sorted key/value pairs; the context manifest of every context used, **transitively**; base and normalisation rules. The manifest digest is a typed digest over its JCS form. *Harness v0.3.0 implements only* `{ registryEntryId, tables, contexts }` *with the contexts named in the table.*
- *Envelope:* harness v0.4.0 uses the seven-element CBOR array `["ssc-m15/v2", profile, registryEntryId, manifestDigest32, cborldBytes, verificationMethod, signature]`; the last two fields are `null` for an inner-proof profile. The parser requires a 32-byte manifest digest. The three-field, 16-byte-digest envelope belongs to the v0.3.0 comparison.
- *Decoder:* recomputes the manifest digest from its local registry entry and context bytes, **before decoding**. On match, it decodes. On mismatch, it looks up the archive by the full 32-byte manifest digest, rehashes the stored manifest and rebuilds it from archived tables and context bytes. It decodes only if that rebuilt digest matches with the verifier's installed decoder and graph-processing profile; otherwise it returns a decoding-manifest-mismatch failure.
- *Binding scope (required):* every input that can affect decoding output is in the manifest, and the decoder loads contexts and codec material **only** from bytes that match it. This is a profile requirement for both forms (DST-9, DST-12); the I-RDFC L2-agreement result itself follows from the authenticated graph even when decoding provenance is incomplete (T10). Harness v0.3.0 pins only the contexts named in the type table; under profile B, a substituted context referenced from inside one of them changes the meaning undetected (X7), and a change of decoder semantics would too (MC-4).
- *Authentication (required):* either the issuer's signature covers the CBOR encoding of the v0.4 envelope's first five fields, `["ssc-m15/v2", profile, registryEntryId, manifestDigest32, cborldBytes]` (profile B: identical inputs to the specified deterministic decoder give identical output), or the payload carries a proof, signed before encoding, that **binds the RDF graph** (`ecdsa-rdfc-2019` in the implemented I-RDFC profile; an LFSC inner-proof profile remains an extension: the verifier uses only the authenticated graph, and no decoded claim before the inner proof verifies). The two profiles give different guarantees; neither binds L3. A signature over the CBOR-LD bytes alone (X4), or an inner proof over the JSON only such as `ecdsa-jcs-2019` (X8), does not protect meaning against a rewritten envelope. WE-2 as published uses an `ecdsa-rdfc-2019` inner proof and resists the rewrite.
- *Manifest v2 and issuer.* The manifest also binds the **graph-processing profile**: the JSON-LD processor, its version and options, which turn the decoded document into a graph. Without it, profile B fixes the decoded JSON but not the graph a verifier derives from it. Every type table MUST be **injective**: two codes for one term, or one code for two terms, would give a single manifest digest two meanings. The issuer is read from the **expanded graph**, not from the compact `issuer` field, which a term alias can bypass: exactly one issuer, equal to the signer. This profile is intended for credentials and enforces exactly one expanded credential-namespace issuer matching the signer; it does not perform complete VC data-model validation. Generic signed documents would need a separate profile. Profile B relies on the integrity of the decoder binary, which is trusted, not measured.
- *Digest length:* 256 bits by default. In an idealised hash model, a second preimage against K target manifests succeeds with probability about $q \cdot K \cdot 2^{-\ell}$, and an adversary who can

influence which manifests exist needs about $2^{\wedge}(\ell/2)$ work for a collision; logging constrains which manifests are accepted but adds no hash bits. Any shorter binding needs a stated threat model and work factor (PRF-3: the complete encoded carrier must fit its capacity).

- *Archive requirement:* manifests, tables and context bytes must be kept, content-addressed by manifest digest, for the product lifetime, under transparency (M9). The harness supplies an in-memory archive record and digest-verified lookup; persistent retention and transparency integration remain deployment obligations.
- *Variants:* signed registry snapshots for offline readers; tables preloaded into a secure element with an M14 rollback counter; for NFC tags, challenge-response with a product-bound key and nonces from an attested source (M13); a carrier PQC transition with a hybrid proof where capacity allows, otherwise a classical proof plus a PQC countersignature held in the archive, which then is mandatory evidence: a verifier that cannot retrieve it after the migration date returns *unknown*, never a classical-only acceptance.

M29 — Trusted lists for publishers of meaning. A trusted-list entry (in the style of ETSI TS 119 612/119 602) for a **semantic publisher** contains: the publisher's identity; the namespaces, credential types or attestation schemes it may publish meaning for; its root keys (M8); its status and history; and the supervisory body. Verifiers at the highest assurance level accept a root role only if it chains to such an entry.

S1.4 Verify

M17 — `ecdsa-lfsc-2026` (experimental). *Experimental profile `lfsc-g/0.4` implemented in `src/lfsc.mjs`; current authorisation, status validation and the variants below require the stated caller or deployment mechanisms.*

- *Proof configuration:* a Data Integrity proof with `type: DataIntegrityProof`, `cryptosuite: ecdsa-lfsc-2026`, `profile: lfsc-g/0.4`, `created`, `verificationMethod`, `proofPurpose`, and:
  - `lexicalDigest`: the typed, domain-separated SHA-256 identifier `typedId(ssc:lexical/v1, JCS(unsecuredCredential))`, encoded as `<type>:sha256-<base64>` by the harness;
  - `semanticCommitment`: `{ canonicalization: "RDFC-1.0", processor, digest: <typed SHA-256 identifier of canonical N-Quads>, dependencies: [ { id, digestSRI } ... ], contextManifest: <typed SHA-256 identifier of JCS(dependencies)> }`; dependency-byte digests use SHA-384 SRI for **every resource the processor dereferenced** while expanding the credential: top-level contexts, contexts embedded anywhere in the document data, type- and property-scoped contexts, and imported contexts, recursively; plus the processor identity and options and the relevant loader metadata (the context manifest, S3.1.3). It is recorded from the loader's requests at issuance. (Harness v0.3.0 lists the top-level contexts only. The binding still holds, because the RDFC comparison detects a substituted dependency, but verifiers must reject undeclared resolutions (VER-16), and receipts and archives need the full list.)
- *Signing input in lfsc-g/0.4:* the proof configuration plus the credential's `@context`, without `proofValue`, is JCS-canonicalised and hashed with SHA-256 for both supported curves. Those 32 bytes are passed to Node's ECDSA signing operation with SHA-256 for P-256 or SHA-384 for P-384. The resulting IEEE P1363 `r||s` signature is encoded as multibase base58btc. This specifies the experimental harness encoding, including both hash steps.

- *Verification:* stage 0, a raw-text I-JSON check for string or Buffer input, then the supplied certified-admission checker (M18); the legacy lexical pre-scan remains the API fallback for comparison. Parsed-object callers must separately establish the strict input domain, and authorising deployments requiring M18 must supply certified admission; stage 1, check the suite and profile identifiers (must-understand), check the proof-value alphabet and maximum encoded length and bound the verification-method length before decoding, then require the decoded signature to contain exactly 64 bytes for P-256 or 96 bytes for P-384, check that the verification method is **authorised for assertion by the issuer named in the credential** (from local data), then the lexical digest and the signature, with no JSON-LD processing; stage 2, resolve only resources listed in the commitment with matching bytes (reject anything undeclared), canonicalise under an operation bound (M19) and compare the typed semantic digest with `semanticCommitment.digest`. No semantic action is authorised before stage 2 completes. *Cache contract (PRF-5):* a positive entry stores the **immutable authenticated graph** and is reused only after the lexical input is re-bound to it, after current dependency bytes and issuer-key authorisation have been checked. The harness key includes `proofValue`, semantic and context-manifest digests, verifier processor and bound, issuer, verification method and epoch; it therefore does not avoid cache misses from equivalent ECDSA signatures. The caller supplies current status and authorisation information and a validated decision time where current validity is required; cache reuse does not establish those facts. Negative and *unknown* entries have bounded lifetime and storage, and never suppress re-evaluation after a dependency changes. (Harness v0.3.0 returns a Boolean on a hit before checking context bytes, so a warm cache accepted substituted context bytes that a cold verification rejects: X13.) An unknown cryptosuite identifier makes legacy verifiers reject the proof instead of skipping the semantic check.
- *Proof type, purpose, time and negative cache.* Stage 1 requires `type: DataIntegrityProof` and `proofPurpose: assertionMethod`, and bounds the length of every encoded key identifier (for example a `did:key`) before decoding it. This experimental LFSC profile requires a `created` timestamp; any supplied `validFrom` and `validUntil` must also have a four-digit year, a valid calendar date and an explicit time zone. Their syntax is checked even when no decision time is supplied; the validity bounds themselves are optional. The profile does not compare `created` with `validAt` or a trusted current time, so proof-creation chronology and freshness are not established. Validity at the decision time is a separate **`validAt` obligation**: an authorising caller supplies a current, validated time and requires `validityChecked: true`; the verifier compares it with the authenticated bounds on every use, including cache hits. A cryptographic-only success does not establish current validity. Graph-native re-verification establishes integrity of the committed dataset, not current authorisation. Negative-cache entries include the secured input and trust epoch, expire, and record only intrinsic stage-1 failures (proof form, digest or signature); admission and issuer-authorisation outcomes are not cached.
- *Security (formal model, T2):* for an honest issuer, acceptance implies, except with negligible probability, that the verifier's RDF dataset is isomorphic to the one the issuer's Issue algorithm computed; for a malicious issuer, only to what it signed. Premises: EUF-CMA signatures, a collision-resistant hash, canonical RDFC, the strict JSON value domain, and a key bound to the named issuer. This is agreement conditional on acceptance, not a guarantee that every processor accepts. It binds layer L2; the meaning of the IRIs (L3) is bound by M3 and M22.
- *Graph-native re-verification:* given stored canonical N-Quads, the stored configuration and the stored issuer, check that the verification method is authorised for that issuer, recompute the N-Quads digest, compare with the commitment, and verify the signature over the configuration. No JSON bytes and no context documents are needed. This re-verifies the

whole committed dataset; standalone proofs for single facts or subsets would need a separate membership scheme. (Harness v0.3.0 omits the issuer check in both paths and decodes `proofValue` in time quadratic in its length before the signature check: X10, X11 (S3.5). v0.4.0 fixes both.)

- *Variants:* the same commitment carried in a JOSE/COSE envelope header instead of a Data Integrity proof; a proof set pairing an `ecdsa-rdfc-2019` proof with a JCS proof whose options carry the RDFC proof's hash; per-statement commitments (HMAC-relabelled statement lists as in `ecdsa-sd-2023`) for selective disclosure, with the lexical proof over the list's root; the commitment combined with an M15 manifest digest for CBOR-LD; EdDSA, or ML-DSA with its own specification and test vectors, instead of ECDSA. *Withdrawn:* an earlier "light" policy that stopped after stage 1 for low-risk actions; stage 1 does not establish meaning, so no semantic action may be authorised before stage 2. *Future work:* the selective-disclosure variant needs a defined completeness rule for policies (VER-8).

M18 — Blank-node-free admission, certified closure and document scan. *v0.4.0 implementation: `src/admission.mjs`, enforcing the restricted ssc-admission/v1 grammar under its bound processor profile, with finite regression evidence and a trusted certifier.* The admission condition is that the expanded credential contains no blank nodes. The formal model (T9.2) lists sufficient conditions: no node object without an identifier (also under `@reverse`, `@included`, `@nest`); no blank-node identifier reached directly or through a term, compact-IRI prefix, `@vocab`, `@type` value or map key; no list (explicit `@list`, or any value of a term with `@container: @list`); no graph container or graph map without `@id`; no `rdfDirection: compound-literal`; and no redefinition or aliasing of `@id` by a scoped context. These conditions are evaluated on expanded IRIs and keyword semantics, in every context reachable from the document. **Responsibility is split.** (1) A **restricted admission grammar** fixes which JSON-LD features may appear. (2) At publication, the publisher's pipeline analyses the compiled closure (M5) and issues a signed **certificate** (PRD-10) that binds the complete closure digest, the processor and its options, the grammar version and the certification procedure, and states that every document admitted by the grammar expands without blank nodes. Until a certificate checker is specified and proved sound, the certifier is a trusted party. (3) At verification, before any JSON-LD processing, the verifier accepts only closures with such a certificate, enforces exactly the certified grammar and options, and checks document-level rules in one pass: size and depth limits; a raw-text I-JSON check (VER-14); keywords from an allow-list only; every node object carries an absolute IRI identifier; no string value begins with `_:`. *Correction:* harness v0.3.0 performs only a lexical scan. Inputs were found that produce blank nodes and pass it: graph containers, list terms in scoped contexts or with non-array values, `id` redefined by a property-scoped context, `@type` and `@vocab`-coerced values naming `_:` terms, compact IRIs with a `_:` prefix (X1, X9); its LFSC path also omits list terms. v0.4.0 implements the split design. Values that can repeat are modelled as explicitly identified members with a position property, not as RDF lists (F5). The proof object is a pinned pattern and is skipped (LFSC never canonicalises it as RDF; a verifier that canonicalises proof options with RDFC must scan them too). Where node identifiers must not be shared across credentials, the issuer derives per-credential skolem IRIs as `urn:hmac:<HMAC-SHA-256(k, JSON path)>`, with a fresh k per credential. These IRIs are **stable across presentations of the same credential** (as with HMAC relabelling in `ecdsa-sd-2023`), so they separate credentials but do not make repeated presentations unlinkable (PRV-1). *Variants:* a small fixed number of blank nodes in pinned structural patterns listed in the certified closure; the same rule for evidence-graph fragments exchanged between companies.

M19 — Deterministic bounds. Expansion, canonicalisation and validation run under an **operation bound** that does not depend on host speed: a count of RDFC hash-n-degree invocations (as `maxDeepIterations` in `rdf-canonize`), a count of quad comparisons, or deterministic fuel metering in a WebAssembly runtime. The bound is a declared parameter of the closure (M1

step 5). Exceeding it yields *invalid: budget exceeded*, identically on every machine. A wall-clock timeout remains only as a liveness guard; its expiry yields *unknown* (escalate or retry), never *invalid*. The receipt records the bound and the operations used. *Variants:* per-risk-class bounds; gas-style accounting per JSON-LD algorithm step.

M20 — No-egress verification cell. Verification runs in a cell without network capability (default-deny sandbox, a WebAssembly component without network imports, or a process with no egress route). Contexts, compiled snapshots, status-list snapshots and trust lists arrive only as signed inputs from a separate fetcher with a host allowlist. The fetcher must not learn which credential is being verified: it runs on credential-independent schedules (prefetch, fixed or padded fetches), and cache misses inside the cell never trigger credential-specific fetches; otherwise the cell leaks through the fetcher (T18, PRV-5). Resolved-context caches are keyed by content digest and scoped per trust domain (never by URL alone, never process-wide across domains). The cell's security-relevant configuration (loader mode, allowlist digest, algorithm allowlist, bounds, blank-node rule, fail-closed policy, must-understand handling) is a **measured configuration manifest**, extended into M24 measurements, so that consumers can accept results only from cells with an approved configuration.

M21 — Must-understand semantics and pinned status profiles. Meaning-changing extensions carry a must-understand marker (for example a `critical` list of extension identifiers in the proof or credential); verifiers reject unknown critical extensions. A status-list profile is a semantic artefact with: purpose semantics; bit order; codec; masking; cover-position lifecycle (per W3C #218); maximum staleness. Verifiers pin profiles by digest and treat stale status as *unknown*. Interpretation-critical members (types, type-scoped contexts, members that carry contexts) are excluded from selective disclosure, or each disclosure yields issuer-authenticated quads (PoS P2). Authentic disclosed statements are not a complete view: a policy declares the statements and types it requires, and a missing required statement yields *unknown*, so a holder cannot hide an authentic restriction (VER-8). Render templates never drive validation; templates used for human decisions are pinned by digest.

S1.5 Decide

M22 — Semantic-closure digest in receipts and admission. Every verification receipt contains: credential identifier and proof digest; the semantic-closure digest (M3) of the meaning applied; a decision-context digest over the reconstructible decision context (S3.1.5), including the verifier configuration with library versions and bound semantics, the policy, status evidence and key and trust-registry snapshots; the authority epoch; the operation bound and its use (M19); the relying party's challenge where one exists; the outcome (*valid*, *profile-invalid*, *cryptographically invalid*, *unknown*). Personal data are referenced through an access-controlled archive rather than copied into the receipt. Agent and policy admission takes the closure digest as an input and refuses actions whose closure is not approved for the current epoch.

M25 — Withdrawal as an authority-epoch event. When a semantic artefact is withdrawn (M8 status, M30 advisory) or a split view is detected (M9), every policy engine or agent safety kernel that holds a mandate or rule whose closure contains the artefact **advances its authority epoch**. Admissions issued under the old epoch can no longer commit. Edge directors sign a new bundle (M14); relying parties stop accepting attested closures that contain the withdrawn artefact (M24, demonstrated in WE-3 E). Push signals (for example OpenID SSF/CAEP) are hints; authority stays with signed metadata.

M26 — Definition-level poisoning of agents. An attacker changes a definition (a class, a context term, a shape) rather than a data instance, so an agent interprets unchanged evidence

differently. Control: agents retrieve evidence only together with closure digests (M22), refuse closures that are not approved for the current epoch, and record them in decision receipts.

M27 — Time-scoped meaning. A decision record contains: the closure valid when the evidence was **certified**, using a validated RFC 3161 timestamp or a trusted bounded log-inclusion time bound to the evidence digest, never the issuer-asserted `created` field alone (this certifies when the evidence existed, not when the underlying event occurred); the closure applicable at decision time; the selection rule used (for example "classify under the HS edition in force at the date of import; map with the WCO correlation table, digest ..."); and the crosswalk digest (M28) if one was used. The record is signed with the decision.

M31 — Compliance-evidence profile. A compliance decision rests on a sufficient evidence set fixed per purpose and decision, with coverage established independently of the holder. The relying party evaluates that evidence under full disclosure, or relies on a certifier's outcome over a committed evidence graph. In the commit-and-certify harness, the custodian performs a second evaluation before acknowledging the deposit (DEC-8 to DEC-15; T19).

The following clauses specify the target profile. WE-7 demonstrates a synthetic subset: fixed declaration fields and source maps, signed complete-list and status responses, evidence-graph commitment, a signed certificate core, local scope/status/policy checks, and deposit/replay using a separately configured instance of the same evaluator. It does not implement independently administered provisioning, full issuer/dependency lifecycle, a complete decision-context archive, periodic recovery, an external effect gate, qualified timestamping, or regulated conformity assessment. Its salt uses Node.js `randomBytes(32)` without an entropy-source certification claim.

WE-7 uses extraction/v2 and accepts only nonempty, well-formed string Map keys whose values are statement objects. No statement body, in the Map, the declaration or the coverage responses, may carry a field named `sourceEntry`: body fields become predicates in the namespace of the committed tuple, so without this rule statements that share another entry's subject could reproduce that entry's tuple, and evidence with and without the entry would have one canonical graph. Each signed statement is committed with its container role and Map key in one canonical `sourceEntry` tuple. Preserving all associations used by evaluation additionally requires a caller premise: statement envelopes and bodies, the declaration and the coverage responses are plain JSON data, with no evaluation-relevant structure lost by serialisation. This excludes accessors, custom prototypes or `toJSON` methods, sparse arrays and named properties on arrays. The harness checks the Map keys, the presence of object bodies and the reserved field name; it does not enforce the recursive JSON-data premise. In the cases tested, certification refused such evidence, because the custodian replays the archived JSON. The synthetic profile rejects missing or multiple matching absence attestations and sorts holder-only report identifiers, so equivalent Map iteration orders yield the same outcome. The extraction change alters the closure and derived policy digests; older certificates require their original extraction implementation. The certifier and custodian are separate instances using the same evaluator and trust-context factory, so configuration separation does not demonstrate independently administered provisioning. Archives contain evidence, coverage, salt, policy and time; executable code, trust anchors, accreditation, profile and mode remain external. The dispute field `contextReconstructible` checks the archived coverage/time binding and policy availability, not complete context reconstruction. The example implements only part of DEC-13 and DEC-14.

Profile (DEC-9). Published as a semantic artefact under M8 and M9. It contains:

- **Identity:** an identifier and version, the subject type, the action, the purpose and the jurisdiction.

- **The sufficient key-claim set K:** required statements as a closed SHACL shape or as property paths with cardinalities, and the identifiers of the policies that consume it.
- **The coverage contract:**
  - the authoritative sources and domains for every policy-relevant dependency, including dependencies reachable through the pinned semantics;
  - source identity and authority;
  - subject and identifier resolution;
  - complete query, pagination or issuer-push rules;
  - snapshot and cut-off rules, with reconciliation of additions, deletions, supersession and revocation;
  - the positive or absence attestations that count, with subject, population or sample, category, method and time, including authenticated selection or cardinality rules for multiple matching attestations;
  - the rule that missing, stale or inconsistent coverage yields *unknown*.
- **Where sampling is used,** its statistical contract: uniformly sampling q of N records misses a single decisive bad record with probability $1 - q/N$.

Regime A: full disclosure of K. The holder discloses every statement of K and may withhold everything outside it. The verifier:

- checks the presentation against the closed profile;
- obtains or checks coverage as the profile requires;
- evaluates the policy;
- records the profile digest in the receipt (M22).

Selective-disclosure formats may be used for policy-irrelevant fields, or for proofs whose sufficiency for the exact policy and closure is established (DEC-8, VER-8).

Regime B: commit-and-certify.

1. **Commit.**
   - The evidence graph G is admitted without blank nodes (M18) and canonicalised with RDFC-1.0. It is built only from validated IRIs and escaped literals, so a value cannot inject statements into G.
   - A 256-bit salt r is drawn from an approved source, by whoever generates it, holder or certifier (KEY-6).
   - h = TypedDigest("ssc:evidence-commitment/v1", r, Can(G)), with r generated as 32 octets. In the harness, TypedDigest hashes `u32be(len(tag))` ‖ `UTF8(tag)` ‖ `u64be(len(r))` ‖ `r` ‖ `u64be(len(UTF8(Can(G))))` ‖ `UTF8(Can(G))` using SHA-256.
2. **Certify (DEC-10).** The certifier receives G and r over a confidential channel and recomputes h. It then:
   - validates each source's authentication, authority and status, and the coverage required by the profile;
   - binds each policy-relevant statement to the identifiers and scope fields used to select it. WE-7 checks a product record's signed identifier, batch and manufacturer; each covered test report's signed identifier, product and batch; and the selected absence attestation's report, product, batch and substance coverage. Absence selection does

not require equality of the Map key and the attestation's own identifier. For inputs that satisfy the JSON-data caller premise above and the enforced checks (nonempty, well-formed string keys, each holding a statement object, and no statement body with a field named `sourceEntry`), extraction/v2 commits each container role, Map key and signed-statement association using a single canonical sourceEntry tuple, so canonicalisation cannot discard input structure used by evaluation. The checks alone do not ensure this for arbitrary JavaScript objects;
- validates every numeric and time field (finite values in range, ordered intervals, no issuance in the future); a missing or malformed value gives *unknown*, never a pass;
- evaluates the policy under the pinned closure.

It signs a canonical **certificate core** with:

- `commitment` (h);
- `semanticClosureDigest`;
- `policy` (identifier, version and digest);
- `outcome`;
- `scope`: subject identifiers, action, purpose, audience and validity period;
- `coverage` (a reference to the coverage evidence);
- `privateContextBinding`: the decision context, with private inputs bound only through h or another hiding binding, never through a plain digest of G (PRV-6);
- `authorization` (the certifier's authority for this decision class);
- `retention` (the archive manifest and custodian);
- a status entry.

Trusted time is an RFC 3161 token or a log inclusion over the digest of the signed core, carried in an envelope beside it, not inside it.

3. **Accept (DEC-11).** The relying party checks:

   - the signature;
   - the certifier's authority for the decision class;
   - that the outcome authorises the request;
   - the scope against the live subject, action, purpose, audience and time;
   - the status of the certificate, issuer and dependencies;
   - that the closure is approved and not withdrawn (M25);
   - that the policy version applies (M27);
   - where one snapshot per scope and epoch is authoritative, the selected certificate under the log's selection rule (DEC-15).

   Missing or stale evidence gives *unknown*. An effect also passes the commit-point check (DEC-7).

4. **Retain (DEC-13).**

   - A conforming custodian acknowledges a **verified** deposit of G, r, the closure and every decision-context dependency. It opens the commitment, checks the context binding and replays under independently provisioned policies and context, refusing mismatch or *unknown*. WE-7 implements these checks for evidence, coverage, salt, policy and evaluation time; trust keys, accreditation, the coverage profile, evaluator and mode are supplied externally. The scenario configures both actors from the same context factory and uses the same evaluator implementation, so it does not demonstrate independent administration or complete archival reconstruction.
   - The acknowledgement binds the certificate identifier, the digest of the certificate core, the commitment, the archive digest and the replayed outcome. WE-7 verifies the acknowledgement signature and compares the certificate identifier, core digest,

commitment and outcome with the certificate. The signed archive digest is retained but is not independently recomputed or compared by the relying party, which does not receive the ciphertext (DEC-11).
- It demonstrates reconstruction on deposit and at the intervals the profile sets.
- Possession or retrievability audits may be used.
- Evidence records renew hashes and timestamps (RFC 4998).
- Access follows defined release rules, such as a court order or an authority request.

5. **Lifecycle (DEC-12).** Certificates are indexed by every decision-relevant dependency: policy, closure, inputs and status. After an invalidating event, including a policy correction that leaves the closure unchanged, current reliance stops within the declared freshness bound until re-evaluation with current inputs succeeds. A replacement links the prior record, which stays replayable as history.
6. **Dispute (DEC-14).** The custodian releases G and r to the court's expert or the competent authority. The report states separately:

   - the authenticity of the certificate and evidence;
   - the validity of the opening;
   - the reconstructibility of the context, checked against the certificate's private-context binding before replay. WE-7 checks its archived coverage/time binding and whether the named policy is available; its `contextReconstructible` field does not establish that the externally supplied trust keys, accreditation, coverage profile, evaluator and mode reconstruct the original context;
   - the equality of the outcome on replay (T14).

   An outcome mismatch shows an inconsistent computation only after the evidence opening and the complete original decision context have both been established; otherwise it may reflect an unavailable or different context. A failed opening does not by itself identify who is at fault. A match does not show that the evidence was true or complete.

Costs (T19.3). Certification requires acquisition, authentication and coverage checks, canonicalisation, hashing and policy evaluation. WE-7 additionally canonicalises and replays at custodian deposit, so successful issuance performs two evaluations. Under the stated graph-size model, canonicalisation costs O(n log n); WE-7 reports scenario outcomes rather than a complexity benchmark. Custodian deposit replay must be charged to archival work where the cost model already includes that term, rather than counted twice. Acceptance does not traverse G when certificate, trust and status data are bounded. Dispute retrieves, opens and reconstructs the archive, canonicalises G and evaluates again.

Variants:

- the certifier is an attested decision engine (M24), or a threshold of certifiers with replay (as in T11′);
- a Merkle commitment over the sorted N-Quads, so that single statements can be opened for operational use. Such partial openings must not feed the compliance decision;
- independent sources jointly evaluating the policy by secure multi-party computation, without exchanging plaintext;
- a commit-and-prove proof that the policy evaluates to the certified outcome on the committed graph. This removes trust in the evaluation, not in the evidence, and has not been implemented or benchmarked here.

Boundary.

- Completeness rests on the coverage contract and on sources that are sound for their declared responsibilities.
- Without a sound computation proof, the relying party relies on trusted certification and custodian replay. Deposit replay precedes acceptance; dispute replay provides a later check of computation consistency.
- Stable certificate and status identifiers are linkable.
- Selective disclosure remains appropriate for small checks on natural persons whose policy is grant-monotone (T19.1).

S1.6 Monitor

M23 — Semantic BOM and drift testing. Verifiers, wallets and agent runtimes report, as signed attestations, the digests of the semantic artefacts they have loaded (including hard-coded mappings, compiled as artefacts). Operators run differential tests: the same corpus through different vendors' verifiers must give identical expanded graphs and outcomes; differences indicate drift.

M24 — Attested decisions: an isolated engine with exclusive key use. *Implemented in part by the v0.4.0 WE-3 software-TPM protocol simulation (`src/attestation.mjs`, `src/engine-process.mjs`, swtpm and tpm2-tools) and the WE-3R two-signature stand-in (`src/replay-receipts.mjs`). The v0.3.0 implementation is a defeated predecessor. Host-root isolation, hardware trust, MON-8 enrolment and threshold signing remain outside the demonstrated harness.*

- *What must be attested.* A register records what a measurement agent was told was loaded. A compromised application can decide in its own code, a reboot clears a PCR-based log, and a key whose policy only requires a PCR value can be invoked by any host caller while that value holds. None of these binds a decision to its meaning (T11; MC-3 V1–V4).

- *Engine.* The decision engine runs as an **isolated runtime**: a TEE enclave or confidential VM, or an OS-enforced isolated process whose memory and state the host cannot modify. Its code is measured. It loads semantic bundles itself, takes an immutable snapshot of input and closure for each decision, decides, and signs a receipt with the relying party's fresh challenge, the recipient or purpose, a decision identifier, the outcome and the context digests (M22).

- *Engine key.* A non-exportable key that **only the engine can use**: generated inside the TEE, or a TPM key whose authorisation the platform grants only to the isolated engine (for example a policy requiring a secret or a signed authorisation held only inside the engine, in addition to the measured state). A policy that requires only a PCR value is **insufficient**. The relying party authenticates and pins the exact key, verifies its binding to the approved engine and policy, and establishes exclusive use and current technical authorisation under T11 (ii-a) and (vi). For current reliance under the high-assurance profile, it additionally requires the current **enrolment record** and actual-use checks of MON-8.

- *Enrolment and attribution (T11 (ii-a), (ii-b); MON-8).* Technical key authentication and exclusive use are required for an execution-bound claim. Organisational attribution is a separate premise, which the high-assurance profile requires for every execution-bound or replay-validated receipt (MON-8). These premises are established separately and by different evidence.

- *Exclusive use (ii-a).* A TPM key certificate (`TPM2_Certify`) shows the key's policy, not who holds its authorisation secret: a key the host creates with the approved policy also passes it (WE-3 v0.4, scenario E3). Exclusive use therefore needs evidence the stated adversary cannot reproduce: a TEE attestation that the key was generated and is used inside the engine, or creation in a controlled ceremony followed by continuing platform protection.
- *Attribution (ii-b).* An enrolment authority, recognised for that function under the ecosystem's governance, issues an enrolment record. It binds the receipt key (for T11′, the group key and each participant's verification share) to the device identity, the approved engine and key-policy versions, the key epoch, the responsible legal person, the authorised decision scope, validity and status. Technical key origin and legal identity and mandate are validated independently, each from a designated authoritative source, and the sources are recorded.
- *Assurance paths.* The profile may use a TPM manufacturer's endorsement and TCG device-identity mechanisms, an IEEE 802.1AR device identity, and an applicable onboarding protocol such as BRSKI or FIDO Device Onboard. These are profile-dependent components, not a mandatory linear chain: FDO does not require an IEEE 802.1AR identity, and device identity does not by itself establish a legal mandate. RFC 9683 concerns TPM-based remote integrity verification. The enrolment service authenticates each selected association, binds the exact receipt key and epoch to the device and approved engine, and independently validates the legal-person mandate and scope.
- *Roles.* An enrolment service may perform the RATS Verifier and Relying Party roles (RFC 9334) and additionally issue application-level enrolment credentials. A RATS Endorser supplies Endorsements about an Attester's capabilities to support evidence appraisal, such as a manufacturer's statement about a protected signing key. Issuing an organisational authorisation is not by itself a RATS Endorsement.
- *Legal effect.* A recognised enrolment authority and a qualified trust service provider are distinct roles, which one entity may hold on separate grounds. Trusted-list status (eIDAS Art. 22) concerns the listed provider and qualified service; it does not itself confer authority for a decision class. If the enrolment record bears a qualified electronic seal, Art. 35(2) gives the linked data presumptions of integrity and correctness of origin. These do not establish substantive correctness or mandate, and do not make later receipts qualified electronic seals. Qualified electronic attestations of attributes follow Art. 45d and Annex V; attestations issued by or on behalf of a public-sector body responsible for an authentic source follow Art. 45f and Annex VII. Article 45b governs their stated legal effects, Art. 45c limits substitution for required electronic identification, and Art. 45e addresses verification against authentic sources.
- *Enrolment profile and owners.* The ecosystem governance authority specifies recognised enrolment issuers and authoritative evidence sources, the required record fields, freshness and revocation semantics, upgrade and re-enrolment, recovery, ownership transfer, group-configuration changes, and a separate historical-verification mode. The enrolment authority validates technical key-origin evidence separately from legal-identity and mandate evidence and records their sources before issuing the record. For an execution-bound claim, technical evidence must establish exclusive key use against the stated adversary; a key-policy certificate alone is insufficient. The verifier evaluates the resulting acceptance predicate for the exact key or group configuration, claimed bindings and requested use, using authenticated status and the profile's time and rollback rules, and rejects an unsatisfied or unresolved predicate. An enrolment record does not itself demonstrate platform isolation, evidence-source truth or a legal mandate beyond the authority's established competence.

- *Lifecycle (MON-5).* Engine upgrades are authorised (for example with TPM2 PolicyAuthorize over publisher-signed reference values) without allowing replay of revoked authorisations. Engine versions can be revoked and are rollback-protected. Register and NV-index lifecycle (definition, deletion, re-creation, restart) is specified.

- *Appraisal.* The relying party verifies the receipt under the certified key, its challenge, and that the named closure is approved and not withdrawn (M25).

- *Bundle events (monitoring).* On load, the engine also extends a register with a typed digest of `{ type: "semantic-bundle", bundleId, version, contextManifestDigest }`, for fleet drift monitoring (M23). These events are monitoring, not decision binding. A context-manifest digest does not cover the vocabulary closure; withdrawal tracking uses the semantic-closure digest (S3.1.3).

- *Correction.* Harness v0.3.0 receipts are produced by the application, bundle events go to the resettable PCR 23, and the appraisal checks only the last event; a device with TPM access resets PCR 23, re-extends only the approved event and is accepted (X6). Our first correction (measured decisions plus whole-log appraisal of application receipts) fails under reboot and in-application decisions. Our second correction (an engine key bound to a non-resettable PCR) fails when root invokes the key (MC-3 V4).

- *Simulation in v0.4.0.* swtpm holds the engine key, and the engine runs as a separate OS process standing in for the isolated runtime. The host-calls-key scenario runs against two policies:

  - a PCR-only policy: a forged receipt is accepted, reproducing V4's defect;
  - a policy that also requires an authorisation secret held only by the engine process: the tested calls without the authorised secret are rejected.

  The launcher starts the engine with an environment containing only the tool search path and selected TPM transport, dropping inherited preload and loader variables. This blocks the tested inherited `NODE_OPTIONS` preload path; it does not authenticate the Node runtime, tool binaries or libraries, PATH contents, or TPM endpoint. Those remain trusted configuration outside the engine-module measurement. The launch regression exercises the actual launcher and requires the child to reach the ready state before checking that the preload did not run. The engine uses a small flat term-resolution rule and returns *unknown* for the explicitly unsupported constructs, including nested contexts; this is not exhaustive JSON-LD subset admission. Scenario E2 counts only when enrolment produced a certified key and the refusal has the expected policy reason.

  The test exercises receipt construction, policy-gated signing and appraisal under an idealised boundary, in which the test adversary has no access to authorisation secrets, protected memory or bypass paths. Ordinary host root is not excluded by such process boundaries, so isolation is not demonstrated; that requires a TEE or a confidential VM.

- *Alternative: replay-validated receipts with a threshold key (T11′).* This claims less than the isolated engine: the **outcome** of an accepted receipt equals deterministic evaluation of its committed, admissible context. Which closure the device executed is not established (MC-3 V6x).

  - *Key.* The receipt key is shared t-of-n between the device's engine and co-signers in separate failure domains, using FROST (RFC 9591) for Schnorr or EdDSA receipts, or threshold ECDSA by MPC for P-256 receipts. The profile names the scheme, its security game and corruption model, the authenticated group key and key epoch, and

the scheme's key-generation, nonce, session, restart and rotation conditions (MON-7). The device counts among the possibly compromised share-holders.

– *Message.* One injectively encoded, domain-separated receipt message:
m = Encode(receipt profile, group-key identifier and epoch, **authority epoch**, **status expiry**, request or challenge, audience and purpose, decision identifier, action and scope, outcome, typed input digest, semantic-closure digest, decision-context digest, admissibility policy). In v0.4.0 the byte-exact encoding is the JCS array emitted by `encodeMessage`: its leading tag and `profile` field are both `ssc-replay-receipt/v2`; its digest uses the separately registered `ssc:replay-receipt/v1` typed-digest domain. Admissibility is encoded as `current-status` or `window:N`.
The authority epoch and status expiry are those of the status evidence the co-signers checked. Receipt verification alone establishes no current authorisation at presentation: every current reliance, including one without an external effect, must apply the applicable epoch, freshness, withdrawal and scope checks at its defined use point. Historical verification must be identified as such. The harness's `acceptReceipt` checks signatures, schema and request binding; its `ProtectedActuator` separately illustrates epoch and expiry checks at an in-memory effect (receipt v1 lacked the signed epoch and expiry).
The decision-context manifest binds each dependency by its role (DEC-1).

– *Co-signer.* Before emitting a usable share, each co-signer:

1. takes an **owned snapshot** of the proposal before any asynchronous check, parses m, authenticates the request, reconstructs the context and recomputes every digest; a proposal changed after authentication is not the one approved;
2. checks **admissibility** against independently anchored state (MON-6):
   * the closure the request authorises, not merely an approved one;
   * the current authority epoch and status, or a declared freshness window stated in the receipt;
   * a declared clock tolerance, with every time and status field present, finite and ordered;
   * device-only observations, which are typed as claims;
3. replays the decision deterministically under the profile's operation bound (T8), and approves exactly m only if the definitive outcome matches. It refuses on *unknown*. The v0.4.0 stand-in executes a fixed JavaScript decision rule without an operation counter; it does not demonstrate this bounded-evaluation obligation. Approvals are bounded in number, expire with the status evidence or a shorter lifetime, are single-use, and are re-checked against the current closure and authority epoch when the share is produced.
4. A conforming deployment logs a replay record for approval and refusal. The v0.4.0 stand-in returns an outcome and retains temporary successful approvals; it does not implement this audit log.

MC-6 shows what each check prevents.

– *Participant configuration and attribution.* The relying party requires an independently authenticated group key, profile, participant configuration and key/configuration epoch, with scheme-appropriate possession and consistency evidence (MON-7; T11′). For current reliance under the high-assurance profile, MON-8 additionally requires an authenticated enrolment record binding that exact configuration to the responsible legal person and authorised use. Participant enrolment does not establish an honest quorum or separate failure domains.

– *Relying party.* It verifies the group key and key epoch, the exact message schema and profile, the signature, the audience and request binding, and the digests. For current

reliance it applies the applicable authorisation, freshness and withdrawal checks at the defined use point; external effects additionally require the atomic commit-point checks of DEC-7. It consumes the challenge for single-use actions. Historical proof verification is a separate result and does not authorise a current use.

  - *Effect gate.* The gate executes only the action the receipt names, once per receipt, and only while the receipt's authority epoch is current and its status expiry has not passed (DEC-7; a pre-release v0.4.0 candidate executed another action, and executed twice).
  - *Availability.* A protected effect waits for a valid receipt and the commit-point check. While co-signers are offline the action is unavailable, not provisionally allowed (DEC-7; MC-6). Progress despite f withholding share-holders needs $n - f \geq t$, plus the protocol's robustness and a working coordinator (PRF-6). FROST does not promise robustness against participant abort.
  - *Confidentiality.* The decision context is encrypted to the co-signers, which run in a TEE, inside the same organisation, or on minimised context. Co-signers on shared infrastructure need a model of common-mode failures before they count as independent.
  - *Degenerate case.* When the relying party is itself the co-signer, it simply replays the decision: no remote attestation is needed, but the relying party sees the context.
  - *Implemented test.* Harness v0.4.0 uses two separately generated ECDSA P-256 software keys, configured in the same fixture, to sign identical receipt bytes; the relying party requires both signatures. The 21 WE-3 R scenarios exercise replay and acceptance rules. The keys are not independently certified or enrolled under MON-8. This stand-in does not implement or test FROST group signatures, distributed key generation or nonce security, or demonstrate separate failure domains.

- *Variants:* confidential VMs or containers with runtime measurement registers (Intel TDX RTMR or equivalent); Linux IMA measurement and appraisal of the engine binary combined with an OS-enforced isolated process; EAT (RFC 9711) as the evidence format; an appraisal policy that maps closures to assurance levels.

M30 — Signed semantic advisories. An advisory is a signed object: advisory identifier; publisher; affected artefact identifiers and version ranges; affected credential types; severity; description; **required verifier action** (`reject`, `escalate`, `re-verify-under` a named closure); fixed versions; publication time. It is logged (M9), referenced from the M8 status of affected artefacts, and triggers M25.

S1.7 Claim register

Claim IDs are `SR-2026-SSC-<label>` and stay stable across versions. Evidence levels: E1 design rationale · E2 worked example · E3 executable model · E4 prototype measurement · E5 field trial or independent replication. Where a design was corrected, evidence is given per version: E4 measured on a defeated or incomplete predecessor is not evidence for the corrected design. "Novel" means *not found* in the searches of 12–13 November 2026 (web and standards search, no systematic patent search), not proven absent.

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
D1	Semantic artefacts and semantic closures as a supply chain with seven stages, eleven objectives, seventeen attack classes (A–Q) and a formal model (SSC-FM v5.2: meaning layers L0–L4, processing-aware dependency closure, typed digests, decision context, adversary classes, objectives as predicates)	Definition, synthesis, partial formalisation	Novel combination	**E3** for the model-checked fragments; proof sketches otherwise; review history in S5	WP (controls without attack model); Fujie & Suzuki (pipeline classes A–C); TUF (E–G for software) → lifecycle and governance classes for meaning, formalised	Completeness of the taxonomy is not shown; T17–T19 (OBJ-1, OBJ-9, M31) are proof sketches without model checks
F1	Signatures that do not commit to context-derived semantics (JCS-based Data Integrity, JOSE/COSE envelopes) leave meaning unbound under context substitution	Empirical and formal result	Demonstration and proof of a known conditional result	E4 + **proof (T1)**	PoS L1(a); Fujie & Suzuki §VI-A (argued) → shown on two implementations incl. Digital Bazaar's	Does not imply that deployments with authenticated external context pins are affected; one use case for the demonstration
F2	A JCS-signed member that no context defines is dropped from the RDF view while the signature stays valid	Empirical result	Demonstration of a known rule	E4, tested processor and options	DI 1.0 §2.4.3 (implementations using JSON-LD processing must error on dropped data); PoS L1 remark	The signature authenticates the JSON member; a conforming consuming transformation must reject the loss
F3	A URL-keyed process-wide context cache defeats per-loader pinning	Empirical result	Security framing of a known issue	E4, `jsonld` versions and cache path tested	jsonld.js #502, #537, PR #552 (open; the shared cache applies to loader results tagged `static`)	One library and one cache path; not all per-loader pins are bypassable
F4	Symmetric acyclic blank-node trees cause super-linear pre-authentication cost	Empirical result	New attack family (search-relative)	E4, versions and bound settings stated	Fujie & Suzuki (cyclic and complete families; an acyclic control) → symmetric acyclic trees	Mitigated by default in `rdf-canonize` $\geq$ 4.0.0; no general complexity novelty claimed

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
F5	Some RDF lists with repeated interior values make RDFC-1.0 super-linear without a bound, and fail under the default bound of current libraries	Empirical result	Partially anticipated (search of 16 Nov 2026): an SDK guard on identical `@list` values [19] and default-bound false positives [1, §III-C7]; the list cause and its default-bound failure not found	**E4**, independently reproduced (X14); two library stacks	RDFC-1.0 complexity note; library work bounds → identifies such lists as a source of both cost and rejection of legitimate input	Tested list family and stacks only; no universal boundary or asymptotic law; availability impact depends on use; Fedify #824 is a related symptom, cause unconfirmed
M1	Semantic build pipeline under SLSA and in-toto with semantic gates	Architecture	Application	E1	SLSA 1.2, in-toto (software)	—
M2	Expansion-level semantic diff that enforces version and identifier changes	Architecture	Novel combination	E1	ROBOT/Bubastis, schema registries, EBSI/eProcurement rules → JSON-LD instance-corpus expansion, RDFC comparison, enforcement at release	Corpus-relative: changes the corpus does not exercise are not detected
M3	Closure manifests (processing-aware context closure, vocabulary closure, interpretation engine) with shape-downgrade controls; semantic-closure digest	Architecture	Refinement	E1	PoS L1(c); RFC 6920; BagIt (RFC 8493); PROV-O → manifest content, typed digest and SHACL severity/openness as meaning	—
M4	Attested cross-serialisation equivalence over a corpus, for declared transformation languages	Architecture	Novel combination	E1	YAML-LD test suite (isomorphism); W3C Data on the Web Best Practices → corpus-wide, attested release gate	Attests test results, not universal equivalence; Type Metadata is not generally an RDF serialisation
M5	Signed compiled contexts for a restricted language, with an equivalence and admission certificate	Architecture	Novel combination	E1	Ruby json-ld preloaded contexts; `addStatic()` → signing, digest addressing, sharing, certification	Active contexts depend on processing state, base and scope; needs a restricted supported language and a certified compiler

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M6	YAML-LD as a source format only	Architecture	Application	E1	YAML-LD §6	—
M7	Machine-generated semantics as labelled, untrusted source with human approval	Architecture	Application	E1	SLSA source track	The profile's own control; AI Act Art. 50 covers specific content categories, not this obligation
M8	Update roles, freshness bounded by the honest status-bearing object, two high-water marks, permanent withdrawal, trusted time	Architecture	Novel combination	**E3** for the modelled sub-protocol (T4–T6; MC-1, MC-5)	TUF/Uptane (software); EBSI TSR; `vct#integrity` → role set and status lifecycle for meaning	**Corrected:** artefact-version counters let replay undo a withdrawal; a fresh timestamp over an old status object breaks a timestamp-based bound. Full TUF role and delegation structure not modelled; some single guards redundant in the instances
M9	Transparency for semantic artefacts with intersecting witness quorums and conflict monitoring; qualified-ledger backend	Architecture	Novel combination	E1 + proof sketch (T7)	RFC 9943 (architecture); witness cosigning; Curty et al. 2022; TU Graz CIT 2024 → semantic statement profile	Fork exclusion needs quorum intersection; detection needs monitors and a conflict relation; after the fact; wire-level profile not yet specified
M10	Provenance-gated signing inside the HSM or TEE boundary	Architecture	Novel combination	E1	Kritis Signer; DiVerify; US 10972277, US 12229267 → semantic provenance checked inside the boundary	—

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M11	Producer-side threshold release only after independent rebuilds match	Architecture	Novel combination	E1	apt-transport-in-toto; repro-threshold (client side) → producer-side share release	A t-of-n signature proves share participation; with f corrupt participants at least t − f honest reproductions; requires distinct administration and binding of shares to build evidence
M12	PQC algorithms assigned by role frequency and lifetime, with mandatory hybrid acceptance	Architecture	Application	E1	NIST SP 800-208; RFC 9964 (ML-DSA for JOSE/COSE); PoS §8.6	A complete transition also needs downgrade prevention, evidence renewal and state handling; an ML-DSA LFSC variant needs its own specification
M13	Attested entropy and status-index assignment by a keyed permutation	Architecture	Novel combination	E1, **incomplete**	Bitstring Status List (random assignment advice); BSI AIS 31 RNG classes → keyed PRP	A key commitment alone proves neither correct assignment nor absence of bias; no audit protocol yet
M14	Edge director with hardware rollback counters and a maximum offline window for meaning	Architecture	Novel combination	E1	Uptane; US 12579272 → semantic bundles, window tied to decision risk	Device identity, re-enrolment and TPM-clear threat model needed before claiming rollback protection across factory reset

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M15	Codec binding outside the compressed payload over a complete decoding manifest, authenticated by an envelope signature or a graph-binding inner proof, with a lifetime archive	Architecture	**Novel**	v0.3: **E4** (drift; RDFC inner proof). Corrected design: **E3** (T10; MC-4 case analysis) and **E4** for the tested v0.4.0 software paths, not a production archive	CBOR-LD 1.0 WD; VC Barcodes WD (sign before encoding); RFC 9124 (SUIT manifests) → explicit decoding manifest, pre-decode check, archive	**Corrected:** unsigned envelopes with byte or JCS signatures, and profile B manifests without transitive contexts or decoder semantics, are defeated (X4, X7, X8, MC-4). +23 bytes is one 128-bit configuration
M17	Lexical-first, semantic-committed proofs with whole-dataset graph-native re-verification (`ecdsa-lfsc-2026`)	Architecture	Novel combination	v0.3: **E4** for the measured paths. Corrected verifier: E1 + **proof sketches (T2, T2.1, T3)** and **E4** for the tested v0.4.0 software paths	PoS L3(iii); `vc-jws-2020`; `eddsa-jcs-2022` → signed configuration with lexical commitment and semantic commitment over the full dependency closure, ordering, scoped caching	Binds L2 (honest issuer) or the signed commitment (malicious issuer), not L3; v0.3 lacks issuer binding, bounded decoding, full closure and the cache contract (X2, X10–X13); light mode withdrawn; not per-fact; experimental suite
M18	Blank-node-free admission through a restricted grammar and a publisher certificate; per-credential HMAC skolem IRIs for cross-credential separation	Architecture	Refinement	v0.3 lexical scan: **E4** for the measured families, unsound in general. Corrected design: E1 + **proof of sufficiency (T9.2)** and **E4** for finite tests of ssc-admission/v1 under its bound processor profile; the certifier remains trusted	PoS A1 (checked on quads) → admission before expansion; certification of the closure	Certifier trusted until a checker is specified; HMAC IRIs do not make presentations unlinkable (X1, X9)

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M19	Operation bounds decide *invalid*; time limits only yield *unknown*	Architecture	Refinement	**E4** (WE-5) for one configuration + **proof (T8)**	PoS §9.1; RDFC-1.0 note; library limits → replayability rule	Determinism needs fixed executable semantics and bound semantics, not operation counts alone; bounds validated on legitimate corpora (F5)
M20	No-egress verification cell; content-addressed, trust-domain-scoped caches; attested configuration	Architecture	Novel combination	E1 (mechanism not built; WE-1 H and I motivate it)	Sandboxing; DI 1.0 §5.14; jsonld.js #552 → attested verifier configuration	Privacy of the separate fetcher needs T18 and PRV-5
M21	Must-understand semantics; pinned status profiles; non-disclosable interpretation-critical members; required statements for policies	Architecture	Application	E1 + proof sketch (T13)	X.509, JOSE `crit`; W3C #218 (proposal); PoS P2	No wire syntax defined; T13 covers extension downgrade only; disclosure completeness needs declared required statements
M22	Semantic-closure and decision-context digests in receipts and admission	Architecture	Novel combination	E2 for partial receipts (WE-3); full decision-context receipts E1 + proof sketch (T14)	Revocation series (receipts); RFC 4998 evidence records; receipts committing to policy, schema and canonicalisation digests [23] → semantic-closure digest as receipt content and admission input	Replay needs reconstructible state and deterministic semantics (T8, T14)
M23	Semantic BOM and differential drift testing	Architecture	Novel combination	E1	CycloneDX data components → runtime inventory of loaded meaning	Monitoring, not proof of which meaning a decision used; corpus-relative

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M24	Decisions bound to their semantic closure by an isolated engine with exclusive use of its signing key. Alternatively, receipts co-signed under a threshold key after admissibility checks and replay, which validate outcomes only. Bundle loads are measured for monitoring	Architecture	Novel combination	v0.3: **E4** for the protocol on a software TPM, **defeated** (X6). Corrected design: **E3** for execution provenance (T11; MC-3 V5) under platform assumptions; replay-validated alternative **E3** for the outcome property (T11′; MC-3 V6, V6x, V6c; MC-6); **E4** for the software-TPM protocol and ordinary two-signature stand-in, not for hardware isolation or threshold cryptography	TEE-bound keys; Linux IMA CRITICAL_-DATA; TPM key policies; RATS/EAT; threshold signing (FROST, RFC 9591); Hyperledger Fabric endorsement; CoSi; threshold-authorised agent actions [24]; BFT-replicated threshold-signed access tokens [25]; quorum-signed oracle reports (Chainlink OCR3) → receipts binding decisions to closures and decision context; co-signers that replay the decision from its committed, admissible context (T11′)	**Corrected:** measured loads, whole-log appraisal and PCR-only key policies all fail (MC-3). Isolation must come from the platform; the v0.4.0 software-TPM protocol simulation does not demonstrate it. The threshold variant does not show what the device executed, fails if t or more share-holders are compromised (V6c), needs admissibility checks (MC-6) and online co-signers. A key certificate does not identify who controls a key; attribution to a legal person needs an authorised enrolment (MON-8), which is a proposal
M25	Withdrawal of meaning advances the authority epoch, checked atomically at the commit point	Architecture	Novel combination (integration with semantic closures)	E4 (WE-3 E, relying party) + **E3** (T12, MC-2)	Revocation series (authority epoch); TUF key rotation	Needs fresh status, permanent withdrawal and an atomic commit point, with external effects fenced; some single guards redundant in the instance

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M26	Definition-level poisoning of agents; closure digests as the control	Architecture	Conjectured attack; partially anticipated (search of 16 Nov 2026)	E1	KG/RAG poisoning (instances); OWASP MCP03 (tool schemas); context poisoning of shared agent contexts, defended by signed versioned contexts [17]; content-hash pinning of agent prompts, tools and configurations [18] → vocabularies, ontologies, contexts, shapes and crosswalks; signed data; the full semantic closure pinned by digest	No measured attack yet; the control stops unapproved changes only, not an approved poisoned definition
M27	Time-scoped selection of meaning recorded in decisions, using certification time	Architecture	Novel combination	E1 + proof sketch (T15)	Bi-VAKs; Temporal RDF; WCO correlation tables; RFC 3161 → signed decision records	Certification time, not event time; the issuer still chooses when to submit
M28	Crosswalks as released semantic artefacts	Architecture	Novel combination	E1	SSSOM; Health-RI governance → set-level closure, signatures, release tests	—
M29	Trusted lists for publishers of meaning	Policy-technical	Novel combination	E1 + proof sketch (T17)	CIR (EU) 2025/1569 Art. 8 (status and version in the scheme catalogue; no withdrawal lifecycle found); ETSI TS 119 602 → publisher accreditation with keys and status	Authenticates authorised publishers, not the correctness of meanings; requires regulatory uptake

ID	Claim	Type	Novelty	Evidence	Closest prior art → difference	Boundary
M30	Signed semantic advisories	Architecture	Novel combination (search of 16 Nov 2026)	E2 for withdrawal behaviour (WE-3 E); advisory format E1	CRA vulnerability handling; VEX; term obsoletion (owl:deprecated, OBO Foundry Principle 19); signed retraction of assertions (nanopublications); trust-list status "withdrawn" (ETSI TS 119 612) → a publisher-signed advisory for terms, definitions and rulebooks, checked at decision time	Format not specified; a profile control, not a statutory duty for every publisher
M31	A compliance-evidence profile: a sufficient evidence set fixed per purpose and decision, with an independent coverage contract. The fixed evidence set is evaluated under full disclosure or by a certifier, with source validation, scope checks, lifecycle, verified archive deposit and replay; deposit and dispute replays are additional evaluations	Architecture and formal result	Integration (search of 16 Nov 2026); the decision-preservation lemma is elementary	E1 + **proof sketches (T19.1–T19.5)**; **E4** for the bounded synthetic WE-7 software example, with partial implementation of the production obligations	omission under selective disclosure [27; 20, §9.3]; SLSA verification summary attestations; in-toto layouts; replay-verifiable conformance receipts with context digests [23]; SD-JWT, BBS and redactable signatures; commit-and-prove (LegoSNARK) and verifiable computation (Pinocchio); zk-creds; Camenisch–Lysyanskaya; execute-endorse (Fabric); WCO Data Model; RFC 4998; PDP and POR → binding of certificates to semantic closure and decision context, coverage contract, lifecycle and archived replay	Completeness rests on authoritative sources and their coverage, not on any disclosure format. The relying party relies on trusted certification and custodian replay unless a sound computation proof is supplied; deposit replay precedes acceptance. Replay shows computation consistency, not truth or completeness. Verifiable computation not implemented or benchmarked here; legal admissibility not established
P1	Requests to EU instruments and standards bodies concerning semantic artefacts (the article, §8)	Policy recommendation	—	E1	PoS §9 (proofs) → semantic artefacts	Our proposals; not positions of the Community Group or any body

M16 (lifetime archive for data carriers) is merged into M15.

Headline evidence. The article's strongest results are the findings F1–F5 (E4, cross-checked; F5 independently reproduced), the proof sketches and the bounded model checks of the formal model (S3). The corrected software paths for M15, M17 and M18, M24's receipt protocols, and M31's synthetic example are implemented and tested in v0.4.0. Their broader architecture and production assumptions remain as stated; M24 execution provenance is model-checked under platform assumptions, and the software-TPM tests do not establish isolation. Historical v0.3.0 performance results remain specific to the earlier, defective implementations. M8 and M25 are model-checked. The remaining lifecycle mechanisms are specified at E1. Five mechanisms (M8, M15, M17, M18, M24) were corrected after independent review (S5); the corrections are stated in the rows above. M31 (compliance evidence) was rebuilt after independent review; its general architecture remains at E1 with proof sketches (T19), with E4 evidence for the bounded WE-7 software example.