HTTP6 technical reading guide

Public technical edition | November 25, 2026

The papers follow a decision from the person who approves a task to the service that records its effect. Start with the technical contribution for the proposed demonstration. Use the specification for component requirements and the review papers for the reasoning, examples, and evidence.

Technical paper	Binder pages
Human-originated agent authority	2 to 5
Complete specifications review	6 to 59
Hereditary authority theory	60 to 70
Payload-bound per-message authorization	71 to 90
Cybersecurity guidance and worked examples	91 to 249

Each paper retains its original byline and printed numbering.

Find accessible HTML chapters and the source files at www.http6.org/standards/nist/.

TECHNICAL CONTRIBUTION / OWNER REVIEW

HTTP6

Technical Contribution: Human-Originated Agent Authority and Payload-Bound Messages

Frank Dylan Rosario; Lin Wang; Octavia Sulea; Francisco Perez; Dawn Lipscomb; Maxwell Calixto; Robert Bloodgood III

Dr. Taher Elgamal — Advisor and contributor

November 10, 2026

NIST NCCoE discussion package — not submitted

Requested action

We request technical feedback on the HTTP6 architecture and discussion of a bounded demonstration for the NCCoE agent-identity and authorization effort. The proposal concerns distinct workload instances, authenticated parent–child provenance, monotonically constrained permissions, and fresh proofs bound to complete application payloads. The Human Admission Client supplies the human-facing admission interface, while the Rosario–Wang construction supplies the private-reference membership component.

The supplied materials consist of the complete available architecture specification, mathematical formalizations, executable review profile, schemas, worked examples, and a reproducible validation kit. The packet identifies the implemented reference profile separately from the native-channel target. It asks for review and suitable project routing; it does not request a generic product certificate or represent an assigned HTTP version.

Contribution and institutional fit

An agent may act long after its user has completed a login and may create several descendant workers. The proposed architecture preserves a bounded origin of authority through that progression. The human authorizes a root scope, the root admits a distinct child under an authenticated policy, and the resource requires a fresh proof for the exact operation. Copying a previous record does not satisfy a new event's state, and a child's local permission change cannot replace the admitted ancestor ceiling.

The NCCoE agent-identity project currently accepts rolling feedback. Its announced first implementation use case is DevSecOps. The project's public summary emphasizes investigating existing standards together with extensions, enhancements, and profiles. We therefore propose a standards-compatible application demonstration while retaining the native HTTP6 channel as a separately specified research objective. [1–3]

The complete source architecture covers service discovery, human and machine admission, independent service identity, handshake transcript binding, encrypted directional records, application request and response semantics, browser/workspace deployment, WASIX capabilities,

policy inheritance, route lifecycle, agent fleets, revocation, and three-tier disaster recovery. Its native cryptographic suite and wire encoding retain explicit release decisions. The supplied local script executes the membership construction and message gate; it is not represented as the complete native browser or TLS replacement.

Proposed bounded demonstration

A developer approves a task with designated repositories, tools, destinations, lifetime, and release restrictions. A protected root admits a scheduler. The scheduler creates separate build and test workers, each with its own key and narrower policy. Workers read approved inputs and write designated artifacts through the resource gate. Publishing to production is a separately approved operation. Human authentication time is recorded separately from machine activity and delegation lifetime.

The delivered executable uses a reporting-worker fixture as its concrete test workload. Adapting it to DevSecOps is proposed work. The demo should compare successful authorized execution with attempted destination substitution, artifact export, copied-instance use, unauthorized minting, concurrent replay, parent revocation, and stale-state recovery. Results should include the actual resource effect, not only a frontend success or rejection message.

Mathematics and security properties

The native response is a private dictionary name of a public membership cell, masked by an independent private residue. Correct verification follows by reversing the mask and dictionary. Calibration and a separating schedule permit authorized reconstruction. Under fresh independent private masks and the specified credential-independent public controls, every supported credential induces the same released-view distribution. Sequential conditional freshness preserves that result for every supported finite observation count, including a trillion. These are source-defined mathematical implications rather than a trillion-exchange benchmark. [4–5]

The payload-bound review envelope adds an independent HMAC over the exact statement and native witness. It supplies computational context authentication. The one-use ledger claims the first eligible authenticated response before native checking. The inherited-policy gate then evaluates current authority before permitting a local effect. Credential privacy, payload authenticity, one-use state, and least privilege are separate parts of the evidence.

The exact mask profile consumes fresh private material. The delivered 24-character RW-40 record uses 150 private mask bits; one trillion such records require 18.75 TB of continuously packed mask material before other overhead. A computational derivation profile requires its own claim and analysis. The submitted source does not convert a fixed-size seed into unlimited independent entropy.

Available executable evidence

A new November 10, 2026 run of the unchanged reconstructed kit passed 81 unit tests, 16 explicit payload scenarios, and two public native vectors. The local flow accepted 18 native proofs for

one admission, one mint, eight requests and eight responses, with eight local effects. A temporary HTTP/1.1 localhost carrier transported three requests and the harness checked three corresponding response proofs locally. The exact finite audit and a 10,000-record primitive benchmark also completed. The environment and exclusions are retained in the raw records. [6]

The kit uses only Python's standard library. Its private setup is represented by protected-role objects in one process; hardware labels are modeled. Synthetic payloads remain readable in the review carrier. The next integration assignments are separately secured endpoints, authenticated private-material provisioning, the selected encrypted transport, real key custody and attestation, a trusted interaction path, distributed durable state, and resource-specific adapters. Reviewers can reproduce the evidence presently supplied and identify the required evidence for each stronger profile.

Review materials and reading order

Material	Purpose
Complete Specifications Review	Current profile separation, exact executable contract, full original architecture with diagrams, mathematical attachment, schema and test index
Updated hereditary-authority theory	Four equations, thirty worked examples, current author credits, and cited SSL history
Per-Message Proof evidence paper	Payload binding, information-theoretic premise, test evidence, resource cost and deployment gates
Expanded NIST review volume	Twelve main chapters and twelve appendices, schemas, reference code and classification/evidence catalogs
Runnable validation kit	Source, 81-test suite, sixteen payload fixtures, two native vectors, one-command reproduction, full READMEs
Mathematical source set	Exact definitions, proofs, worked constructions, observer model, entropy accounting, and Lean declaration registry

The NIST mapping is a proposed engineering crosswalk. Its twenty-family and 106-outcome example coverage is not a claim that all NIST clauses have been assessed. A selected boundary, authoritative requirement inventory, parameter values, provider assignments, and actual assessment establish deployment satisfaction. Likewise, an underlying cryptographic module's validation is distinct from review of the complete protocol. [7]

Questions for the project team

We seek guidance on the most useful bounded use case, expected interoperability with existing agent identity mechanisms, evidence formats, relevant comparison configurations, and the appropriate participation mechanism. We also request guidance on routing a focused review of the private-reference construction and its composition with application authorization. The accompanying native-profile decision register identifies the exact additional deliverables needed before an independently interoperable native-channel claim.

References

[1] NIST NCCoE, agent identity and authorization project page, rolling feedback. Checked November 10, 2026. https://www.nccoe.nist.gov/projects/software-and-si-agent-identity-and-authorization

[2] NIST, September 29, 2026 agent identity announcement, DevSecOps use case and project contact. https://content.govdelivery.com/accounts/USNIST/bulletins/42d04b7

[3] NIST NCCoE, *Summary of Comments on the Concept Paper*. https://pages.nist.gov/nccoe-ai-identity/summary-of-comments.html

[4] *Rosario–Wang: Consolidated Construction*, September 16, 2026, §§3–10; included source.

[5] *HTTP6: Executable Rosario–Wang Proof and Observer Demonstration*, November 9, 2026; included attachment.

[6] `06_evidence/regeneration_2026-11-10/validation_report.json` and corresponding logs; new reference-model execution.

[7] NIST CMVP FAQs, SG-6, protocol versus module validation. https://csrc.nist.gov/projects/cryptographic-module-validation-program/faqs

REVIEW SPECIFICATION / 2026

HTTP6

Complete Specification Set and NIST Technical Review Dossier

Frank Dylan Rosario; Lin Wang; Octavia Sulea; Francisco Perez; Dawn Lipscomb; Maxwell Calixto; Robert Bloodgood III

Dr. Taher Elgamal — Advisor and contributor

November 10, 2026

Review compilation 2.0 — prepared, not submitted

Architecture, mathematical sources, exact reference profile and reproducible evidence.

Prepared for technical review. Native-profile decisions and evidence are identified separately.

Contents

Part I. Review specification and release identity **3**
I.1. Purpose and requested review . . . 3
I.2. Evidence classes and profile boundaries . . . 3
I.3. What the submitted construction establishes . . . 4
I.4. Participants and responsibility boundaries . . . 5
I.5. Exact executable review profile . . . 6
I.6. Complete specification coverage and delivery map . . . 9
I.7. Reproduction and newly executed evidence . . . 10
I.8. Deployment completion decisions . . . 11
I.9. NIST alignment and a bounded demonstration . . . 12
I.10. Source precedence and preserved distinctions . . . 13
I.11. How to use the complete payload . . . 13
I.12. Reference register for this compilation . . . 13

Part II. Complete original HTTP6 architecture **15**
II. Introduction and design thesis . . . 15
II.1. The architectural question . . . 16
II.2. Names, products, and technology layers . . . 17
II.3. Trust principles and core protocol invariants . . . 17
II.4. What changes from the conventional stack . . . 19
II.5. The exact page-to-asset lifecycle: fourteen protocol phases . . . 19
II.6. The opening human proof and synchronized twin oscillation . . . 22
II.7. Session lifetime and route transitions . . . 22
II.8. HTTP6 protocol model, roles, and records . . . 23
II.9. Client/server APIs, headers, and errors . . . 26
II.10. The HTTP6 browser and WASIX deployment architecture . . . 28
II.11. The Wasmer/WASIX service stack . . . 29
II.12. Suggested source modules and repository boundaries . . . 31
II.13. Human and automated authorization families . . . 32
II.14. Implementation work packages and direct deployment path . . . 37
II.15. Operational security, monitoring, and recovery controls . . . 38
II.16. Open engineering decisions to fix in the protocol profile . . . 39
II.17. Architectural conclusion . . . 40
II. Appendix A. Vocabulary and source basis . . . 40

Part III. Executable Rosario–Wang attachment **42**
Abstract . . . 42
A. The implemented statement and private witness . . . 42
B. Geometry selection and the source profiles . . . 43
C. Private entropy as a counted circuit resource . . . 43
D. The observer’s posterior, including public code . . . 44
E. Enrollment, origin, and application authentication . . . 45
F. One-use validation and the local business effect . . . 45
G. Human origin and per-message machine proofs . . . 46
H. Efficiency and quantified experiments . . . 46
I. Reproduction and evidence status . . . 47
Source register . . . 47

Appendix A. Complete executable-envelope schema **48**

Appendix B. Payload-case acceptance register **52**

Appendix C. Schema, model, and source-code inventory **53**

Appendix D. Submission and authorship controls **54**

Part I. Review specification and release identity

I.1. Purpose and requested review

HTTP6 is proposed as a secure application-channel architecture for human, workload, and agent traffic. The Human Admission Client supplies the human-facing approval interface; Rosario–Wang supplies the private-reference membership construction; the Proof Authority, the WASIX execution sandbox, the Policy Decision Point, and the Authoritative State Plane supply instance admission, protected execution, policy, and authoritative state. The review question is whether these components can preserve bounded human-originated authority through every protected request, response, and descendant process. The complete architecture is reproduced in Part II, and the executable mathematical attachment is reproduced in Part III. [D1–D4]

This edition assembles the complete **available specification set**, rather than silently promoting a design objective into an implemented transport. It provides the full architecture, an exact account of the delivered review implementation, source-defined privacy and recovery propositions, deployment acceptance criteria, schemas, source code, and reproducible test evidence. Source manuscripts, reference implementations, and proposed native-profile decisions remain individually identified. The original files are retained separately with SHA-256 preservation records. [D1–D8]

The requested first NIST action is technical feedback and discussion of demonstration fit with the NCCoE agent-identity and authorization project. That project currently reviews feedback on a rolling basis. Its announced initial implementation use case is DevSecOps, and its summary describes evaluating existing standards with extensions, enhancements, and profiles. A bounded agent workflow over an established carrier is therefore the proposed demonstration entry point; the native HTTP6 channel remains the parallel protocol research objective. [N1–N3]

The release date is **November 10, 2026**. The current author group is Frank Dylan Rosario, Lin Wang, Octavia Sulea, Francisco Perez, Dawn Lipscomb, Maxwell Calixto, and Robert Bloodgood III. Dr. Taher Elgamal is credited separately as advisor and contributor, following the supplied revised manuscript. Historical documents keep their original bylines. Author approval, affiliation, contact details, and public-release permissions are recorded in the submission checklist rather than inferred. [D4]

I.2. Evidence classes and profile boundaries

The packet uses four evidence classes. **Mathematical result** means a proposition proved under its stated finite-model hypotheses. **Executed reference result** means an observed behavior of the identified code and test environment. **Architecture requirement** means a target behavior specified for an implementing component. **Release decision** means an algorithm choice, deployment parameter, approval, or review deliverable that must be resolved for the selected profile. These classes permit affirmative, precise claims without confusing their supporting evidence.

Profile	Definition in the supplied material	Evidence in this delivery
Review implementation	`http6-rw-review-1.0`; finite private tape, calibrated native witnesses, independently keyed HMAC envelope, local registry and SQLite	Runnable source, exact arithmetic audits, regression tests, and fresh local/loopback evidence

Profile	Definition in the supplied material	Evidence in this delivery
W1 compatibility	Proof-bearing application records carried through HTTPS; protected resource adapters and the selected inner protection	Full architecture and interface contracts; the delivered plaintext loopback harness is not labeled a completed W1 deployment
W2 managed client	Trusted workspace or native companion, controlled rendering, protected state, explicit carrier	Browser and runtime architecture plus acceptance work packages
N1 native	Independently authenticated service and client instances, transcript-confirmed key establishment, encrypted records, state and lifecycle	Complete architectural target and a decision register for the exact native suite, wire codec, and implementation evidence

The review implementation uses a new native membership word for each demonstration request and response. The broader architecture also permits lightweight record authenticators generated by a previously admitted runtime. These are two explicitly separated record-proof profiles. Neither forces a human to repeat a full login for every image or data chunk. Record class, proof profile, and inherited scope must be fixed in an eventual interoperable specification. [D1, §§6, 8.3; D2, §§2–4]

The native cryptographic schedule is computational: it derives directional keys from authenticated secret inputs and handles key updates. The exact credential-privacy profile separately consumes fresh independent private masks. A key-derivation function does not silently replace that fresh-mask premise. Public timestamps index context; they are not credited as secret entropy. This distinction governs all uses of the phrase “entropy pool” in the compiled packet. [M1, §§7–9; D1, §8.2; D3, §C]

I.3. What the submitted construction establishes

Let $X = (k, \varphi)$ be the enrolled string and private naming bijection, c_t the public membership function, and m_t the designated private mask. The source response and verification relation is

$$w_t = \varphi(c_t(k_{r(t)})), \qquad z_t = (w_t + m_t) \bmod H. \tag{R1}$$

The index $r(t)$ selects the enrolled position. The cell count H is the response modulus. Verification subtracts the actual mask and reverses the naming map, then compares the result with the enrolled membership value. Honest correctness follows by cancellation. Calibration identifies the private dictionary to an authorized decoder, and separating facets identify the symbol. The native verifier and the masks-only recovery routine are separate executable functions. [M1, §§3–5; D3, §A]

When each response coordinate has an independent uniform private mask, the likelihood of every released word is the same for every supported credential. With credential-independent public challenges, and conditional freshness over the full history,

$$\Pr[Z_e = z \mid X = x, B, G_e = g] = H^{-q_e}, \qquad I(X; \mathcal{T}_{1:N} \mid B) = 0. \tag{R2}$$

Here B is the declared prior background, G_e the complete public challenge, q_e the calibrated response count, and $\mathcal{T}_{1:N}$ the specified released history. Exactly one mask explains a fixed output for each candidate credential. Multiplying credential-independent conditional probabilities proves the sequential statement for any supported finite N. The observer's posterior remains its prior, including a nonuniform prior. The proof does not require enumerating a trillion sessions. [M1, §§4, 9]

The observation boundary includes public code and public circuit descriptors. Actual masks and enrolled references are private resources. Supplying those masks to the authorized recovery function recovers the enrolled material under calibration and separation. Acquisition of a secret-bearing runtime is therefore a different exposure experiment from acquisition of the public binary code or descriptor. Timing, errors, device signals, and active feedback are analyzed jointly when included in a stronger observation claim. [M1, §9.3; M2, §§4–6; D3, §D]

For the actual reference envelope, let s_e be the canonical statement and Z_e its native witness. A separate independent key authenticates the complete record:

$$\begin{aligned} C_e &= \{\texttt{statement} : s_e, \texttt{released} : Z_e\}, \\ \tau_e &= \mathrm{HMAC\text{-}SHA256}_{a_v}\left(\begin{array}{c} \mathrm{ASCII}(\texttt{witness-envelope}) \parallel \texttt{0x00} \\ \parallel \mathrm{canonical}(C_e) \end{array} \right). \end{aligned} \tag{R3}$$

The statement contains the actual payload, both peers, channel, direction, stream, epoch, sequence, policy and lineage identities, profile, challenge, and expiration. Changing any covered byte changes the authenticated object. Context authenticity is computational and is additional to the credential-privacy theorem. Application payload confidentiality belongs to the selected encrypted transport/record profile; the runnable local carrier deliberately displays synthetic data. [D2, §§2, 5; D3, §E; D5, `mac`, `Peer.respond`]

I.4. Participants and responsibility boundaries

The human participant approves initial scope through the selected protected admission ceremony. The admission verifier checks the enrolled relation and binds acceptance to the proposed root. A parent instance may ask the protected mint for a child only when its policy permits spawning. A child is a distinct executing principal, with its own reference and key, rather than a copy of the human's identity. The service twin checks incoming records and generates independently protected replies. The resource gate controls the actual effect. These roles are architectural responsibilities, even where the review program represents several of them inside one process. [D1, §§2–3, 8; D2, §2]

A trusted service-discovery relationship must exist before a human is asked to approve a purported service. The enrolled descriptor establishes the intended service identity; routing addresses locate it. Device evidence supplies independently appraised facts about the client key and executing instance. A processor number, interface address, or caller-supplied machine label is a context attribute rather than proof of secret-key possession. The deployed key-use interface also enforces the intended operation before generating its authenticator. [D1, §§5, 8.4; D4, §§5–7]

The confidentiality of a human's commitment does not itself authenticate a child or prevent live forwarding. Heredity is established by the authenticated grant chain, current instance possession, and independent policy gate. A forwarded authentic encrypted record may reach its legitimate endpoint without granting the forwarder permission to alter it. A separate hardware-bound profile checks the local trusted screen, protected confirmation, device key, and exact session against relayed-screen attempts. The current script exercises logical context mismatch; it does not manufacture hardware evidence. [M1, §12.4; D3, §E]

I.5. Exact executable review profile

I.5.1. Encoding and bounded inputs

The preserved program uses `VERSION="http6-rw-review-1.0"`, `MAX_PACKET=1_000_000`, and `MAX_WORD=1024`. Its canonical codec accepts ASCII-encoded JSON produced with sorted keys, separators `,` and `:`, ASCII escapes, and no floating-point values. Null, strings, booleans, bounded exact integers, arrays, and string-keyed objects are supported. Integer magnitude is less than 2^{256}; recursion depth is at most 32; the complete canonical record is at most one million bytes. Wire parsing rejects duplicate keys and any byte sequence that does not equal the canonical re-encoding. This is the program's local codec, not an assertion of RFC 8785 compliance. [D5, `canonical`, `parse_packet`]

Operation, resource, and destination are required nonempty strings of at most 256 characters in a challenge payload. Other payload data must satisfy the canonical codec. Nonces are nonnegative integers below 2^{128}. Default challenge lifetime is 120 seconds; the challenge API accepts lifetimes from 1 through 3,600 seconds. A review runtime's default private tape is 8,192 bytes per enrolled sender. The profile's complete typed envelope schema appears in Appendix A. Schema validation supplements, rather than replaces, the runtime's cross-field and state checks. [D5, `Runtime.challenge`; D6]

The complete statement is immutable once issued. The verifier retains its authoritative copy. Required field groups are shown below; the original schema fixes all field names.

Group	Statement fields	Required interpretation
Profile and event	`protocol`, `kind`, `epoch`, `event`, `nonce`	Exact profile and full event namespace; modulo geometry alone is never the uniqueness key
Peers and context	`sender`, `recipient`, `channel`, `direction`, `stream`	Match independently supplied local binding, not an untrusted header assertion
Order and time	`sequence`, `issued_ns`, `expires_ns`	Fresh current-generation position and unexpired challenge
Instance and policy	`device`, `measurement`, `policy_digest`, `lineage_digest`	Match the registered reference-model instance and unchanged admitted policy
Application	`operation_id`, `request_ref`, `payload`	Complete action bytes and, for a response, the exact prior accepted request
Membership	`profile`, `length`, `calibration_cells`, `responses`, `frames`	Correct geometry, expected calibrated count, and valid residue domains

I.5.2. Geometric profiles and entropy allocation

`ring4` uses symbols `ABCD`, two cells, two data coordinates per symbol, and two calibration entries. `rw12` uses `123456789*0#`, four cells, the twelve unequal ordered pairs in $\mathbb{Z}_4^2$, and three calibration entries. `rw40` uses the source's forty balanced triples with 36 alphanumeric live symbols and four non-enrollable pads. It uses three data coordinates per symbol and three calibration entries. The added cyclic nonce reindexing for `rw40` is explicitly labeled an engineering extension that preserves the table's separation and balance. [M1, §§6, 10; M3, §15; D2, §3]

`PrivateTape.allocate` reserves a complete event bundle under a lock, advances its cursor before releasing any values, and refuses exhaustion or a repeated event allocation. It consumes disjoint bit positions, including allocations later rejected or cancelled. The reference uses operating-system cryptographic randomness to initialize the finite tape. The exact probability audit separately enumerates the ideal distribution. Erasing buffer positions in Python is recorded as an implementation action, not an assurance of physical memory erasure. [D5, `PrivateTape`]

Profile	Enrolled length	Calibrated coordinates	Fresh mask bits per record	Trillion-record packed mask material
RW-12/4	12	27	54	6.75 TB
RW-40/4	24	75	150	18.75 TB

A request and response require two independent record allocations. The table uses decimal terabytes and excludes overhead, duplication, provisioning, and payload encryption. A 65,536-bit tape contains 1,213 complete 54-bit records or 436 complete 150-bit records. The capacity command calculates these quantities symbolically. These are resource requirements of the selected independent-mask encoding, not storage claims for every possible cryptographic profile. [M1, E4–E5; D3, §C]

I.5.3. Challenge, witness, and validation algorithms

The following sequence summarizes the actual code. The full source is included unchanged in the runnable kit.

```
ISSUE_REVIEW_CHALLENGE(binding, payload):
    validate kind, lifetime, payload and canonical encoding
    require registered sender and live ancestry
    select unused full event and next directional stream sequence
    bind exact action, peer, instance, policy, lineage and geometry
    allocate a complete fresh private mask bundle
    store the authoritative issued statement
    privately provision matching sender and verifier mask slots
    return statement and domain-separated authenticated challenge

RESPOND_TO_CHALLENGE(challenge, local_binding, intended_payload):
    authenticate the challenge under the independent instance key
    require exact local peer/channel/direction/stream and intent match
    require registered device, measurement, policy, lineage and profile
    require expected geometry, lifetime and unspent local material
    mark local event spent before computing its response
```

```
    generate calibrated native Rosario-Wang masked word
    authenticate {statement, released} under witness-envelope domain
    return the complete envelope

VERIFY_AND_APPLY(envelope, independent_binding):
    validate representation and exact authoritative statement match
    require expected endpoints, current epoch and valid deadline
    verify witness-envelope authenticator
    atomically claim ISSUED -> BURNED; refuse any repeated claim
    obtain designated private masks and check the native witness
    recheck live ancestry, committed policy and advancing stream order
    require every ancestor's complete-operation policy
    activate root, admit an approved child, validate response linkage,
        or execute/reconcile one local business operation
    record outcome; expose only the permitted public observer record
```

Authentication of the envelope precedes the claim. Native witness verification follows the claim. Thus a malformed or unauthenticated packet does not obtain a native attempt, while an authenticated complete response with an incorrect native word consumes that event and is rejected. Repeating it encounters consumed state. This ordering is a specified part of the source burn model and the actual implementation. [M1, §8; D5, `Runtime.submit`]

I.5.4. One-use state and application outcomes

The mathematical event model has RESERVED, ISSUED, and BURNED stages. The local ledger records an event as ISSUED after its private allocation and uses a committed SQLite transaction for the ISSUED-to-BURNED claim. BURNED is absorbing for the same event identity. Reopening the runtime retires unfinished issued events to `RECOVERY_HOLD`. Private references and masks are not restored from the effect database; this is controlled local restart behavior rather than a whole-machine rollback defense. [M1, §8; D5, `Ledger`]

A business operation is keyed by `(sender,operation_id)`. The first permitted request inserts its canonical payload and event reference in the local effect table. A fresh valid proof for the same operation and identical payload yields `RECONCILED`; the same identifier with different content yields `OPERATION_CONFLICT`. This establishes one local table insertion. An external payment, file write, or database service needs its own durable idempotency and recovery contract at the actual effect boundary. [D3, §F]

The local sequence rule is strictly greater than the last accepted value for that tuple of sender, recipient, channel, direction, and stream. It is not a sliding-window native transport: later acceptance can make an earlier in-flight record stale. The native architecture separately calls for bounded in-flight windows and authenticated predecessor state. The compiled specification retains that distinction rather than attributing transport-level retransmission or reordering support to the reference counter. [D1, §6; D5, `Runtime.submit`]

I.5.5. Peer grants and inherited policy

The reference `Policy` permits exact operation/resource/destination tuples and applies wildcard-capable denials. A registry grant authenticates the instance, parent reference, policy, modeled

device and measurement, and human receipt using an independent local issuer MAC. The new child must have a narrower allow set and a smaller remaining depth; each ancestor's denials and live status are rechecked. Public digital-signature verification, the richer IAM/RBAC schema, shared monetary budgets, attestation, and distributed revocation remain separately identified architecture integrations. [D5, `Policy`, `Registry`; D7]

The demo human fixture can ignite the inactive root. The root can mint `reader-v1` and read the approved report route. A successful mint creates `worker-1` with an independent machine reference and key. That leaf can read the approved resource, cannot export, and cannot mint a grandchild. Root activation records the human receipt; it does not derive child credentials from the human's secret. [D2, §2; D5, `setup_runtime`, `ignite_and_mint`]

I.5.6. Request and response contract

Every REQUEST produces a fresh native witness and envelope. Every RESPONSE uses the service's independent reference and masks. Response validation requires an accepted or reconciled prior REQUEST with matching reversed sender and recipient and the same channel. The response's payload is separately authenticated. A transport response indicating a verifier decision is not itself the modeled service-response proof. The original loopback harness sends requests through a localhost HTTP server and validates the fresh service-response proof locally. [D2, §2; D5, `run_demo`, `Runtime.submit`]

The optional carrier accepts `POST/http6-review/submit` on a temporary port bound to `127.0.0.1`. It requires a bounded Content-Length, rejects Transfer-Encoding, parses canonical bytes, and uses a handler-bound `Binding`. It returns a canonical JSON decision with HTTP 200 or HTTP 400 for handled request errors. The local client disables ambient HTTP proxy use. Only synthetic data is transmitted; no service named `reports.example` is contacted. [D5, `loopback_submit`]

I.5.7. Errors and observation records

The reference emits structured states including ACCEPTED, RECONCILED, REPLAY, EXPIRED, REJECTED, PRIVATE_STATE_UNAVAILABLE, OPERATION_CONFLICT, and RECOVERY_HOLD, with expected validation exceptions recorded by the test harness. The native architecture's proposed error families are retained separately in Part II, §9. The two sets are not asserted to be identical wire enumerations.

`observer_honest.jsonl` contains accepted public statements, masked words, and tags. It intentionally excludes enrolled words, dictionaries, raw responses, tapes, and keys, but includes the synthetic payload and metadata. Failure diagnostics and timing are separate evidence. The exact privacy theorem is applied to its declared view; unrestricted active attempts, complete Python timing, and later private-state exposure are not relabeled as part of the exact finite audit. [D2, §§2, 5]

I.6. Complete specification coverage and delivery map

The following index identifies the entire available design, code, and acceptance basis. Architecture-only rows retain their completion gates in the decision register; every row is included even when it is not exercised by the local script.

Area	Specification source	Delivered implementation or review artifact
Product boundaries and terms	Part II §§1–2	Complete source; Human Admission Client / reference architecture / HTTP6 separation
Trust invariants	Part II §3	Ten architecture invariants and local enforcement correspondence
Service discovery and first contact	Part II §5, phases 1–3	Descriptor requirements; native decision ND-01
Human and machine admission	Part II §§5–6, 13	Synthetic admission fixture and independent child references
Native authenticated handshake	Part II §§5, 8.2	Transcript/key-confirmation requirements; ND-02
Wire encoding and extension rules	Part II §8.1; Part I §5.1	Exact review codec and schema; native codec ND-03
Payload-bound records	Part I §5; Part III §E	Native RW word, HMAC binding, positive/negative payload cases
Streaming and media	Part II §§8.3, 11.2	Chunk/track requirements and performance acceptance gates
Runtime and route identity	Part II §§7, 8.4, 13.1	Logical labels/checks; hardware and route appraisal ND-05
Browser and content handling	Part II §10	Three-stage workspace/native plan; browser gate ND-09
WASIX service placement	Part II §11	Capability boundary, components, and module inventory
Hereditary policy and IAM/RBAC	Part II §13; D7	Full architectural schema plus smaller executable policy
State, retry, and recovery	Part II §§7, 13, 15	SQLite one-use and reconciliation tests; distributed gate ND-06
Offline and catastrophe recovery	Part II §13.2–13.5	Three recovery tiers; fresh epochs and custody requirements
Observer privacy and entropy	Part III §§A–D	Formalizations, exact audits, capacity reports
Security experiments	Part II §8.5; Part III §§E–H	Sixteen explicit payload cases and further acceptance plan
NIST relationships	Assurance volume Chapters 6–10	Twenty family cases, 106 CSF outcomes, classification fixtures
Review evidence and reproduction	Part I §7; Appendix B	Full runnable kit, schema catalog, manifests, logs and instructions

I.7. Reproduction and newly executed evidence

On November 10, 2026 the unchanged reconstructed validation kit was extracted and executed under Python 3.13.5 on Linux, with SQLite 3.46.1. The run passed **81 unit tests**, **16 explicit payload cases**, and **two published native vectors**. The local workflow admitted 18 native proofs and produced eight application effects. The loopback workflow performed three request/response pairs with three effects, under the carrier scope described above. The finite audit checked 2,048 binary executions, 294,912 RW-12 executions at one nonce, 2,048 two-event executions, and 864 calibrated RW-40 single-symbol recoveries. [E1]

The new run also measured 10,000 native primitive records. Its raw benchmark names platform, durations, and exclusions. It does not include the HMAC envelope, database, network, hardware, transport encryption, or private provisioning. Historical benchmark values in preserved papers remain historical and are not overwritten with a differently scoped run. The evidence directory contains complete logs, actual public records, and independently inspected SQLite effects.

From the top level of the extracted delivery:

```
python3 verify_delivery.py
python3 run_validation.py
```

The second command checks the top-level manifest, invokes the unchanged kit, and writes fresh outputs under a new `validation_runs/` directory. On Windows, use `py-3` in place of `python3`. Python 3.10 or newer and permission to bind temporary localhost sockets are required. No external network, API key, user account, or third-party package is required for this primary validation path. Restoring old proof files is not a validation shortcut: the runner creates new challenges and unused private allocations for every new trial. [D2; E1]

The expanded assurance volume also contains a separate signed-authority reference model, finite models, schemas, and an evidence-record checker. Its own dependencies and run instructions remain in that standalone directory. Its recorded original counts are preserved as inherited evidence unless a distinct new receipt explicitly identifies their execution. No Lean kernel build is asserted in this regeneration. [D8; M4]

I.8. Deployment completion decisions

The native profile reaches an interoperable release when the following assignments have concrete specifications, test vectors, and evidence. These are source-defined completion items, not silently selected algorithms or fabricated deployment results.

Decision	Required deliverable	Evidence for closure
ND-01: trusted discovery	Exact descriptor syntax, trusted issuer/bootstrap, key rotation and withdrawal	Independent first-contact, substitution, expiry, and recovery tests
ND-02: native crypto suite	Selected authenticated key establishment, identities, transcript, key confirmation, AEAD, labels and rekey	Complete composition analysis, known-answer vectors, independent endpoint agreement
ND-03: native wire format	Canonical binary codec, framing, maxima, extension rules, negotiation and failure behavior	Byte-identical encoding and adversarial parser corpus
ND-04: private-resource provisioning	Assessed private resource source, delivery, quotas, custody, retirement, and replenishment	Source assessment and leakage/lifecycle evidence for the selected exact or computational claim
ND-05: local hardware and screen	Real quote format, evidence appraisal, protected key operations, trusted display/input and pairing	Live relay, copied-runtime, wrong-device, and misleading-intent experiments
ND-06: distributed state	Consistent allocation/consumption, fencing, replica failover, rollback and durable outcomes	Fault-injection and resource-side results under the declared failures

Decision	Required deliverable	Evidence for closure
ND-07: application adapter	Exact normalization, permission semantics, data classification and idempotent effect integration	No bypass route; authorized effects and prohibited effects independently inspected
ND-08: child identity	Public issuer/signature profile where needed, templates, budgets, lifetime and revocation	Cross-verifier grants, privilege-expansion refusals, shared-budget and cessation tests
ND-09: browser	Origin model, renderer isolation, resource interception, downloads, scripts and accessibility	Browser conformance, protected interaction and usability evidence
ND-10: streaming	Chunk boundaries, ordering windows, ACK meaning, retransmission, close and congestion integration	Throughput, latency, loss, removal and rekey tests
ND-11: recovery	Surviving trust anchors, quorum where selected, epoch transition and stale-state exclusion	Whole-fleet drills, current-checkpoint evidence and measured cessation/recovery
ND-12: assessment and release	Selected NIST inventory, parameters, enforcers, actual deployment, independent reviewer, approvals	Complete requirement records, signed approvals and applicable module evidence

“Complete” in the delivery title refers to the assembled available review corpus and its explicit interfaces. A native implementation-conformance statement will name the resolved N1 profile and its evidence. This permits reviewers to evaluate the full proposal now and identify the exact remaining release contract without treating an omitted choice as an implemented feature.

I.9. NIST alignment and a bounded demonstration

The retained NIST mapping covers all twenty SP 800-53 control families and all 106 CSF 2.0 Core outcomes through source-derived mechanisms or expressly proposed integrations. It also includes separate identity, cryptographic-module, CUI, privacy, and lifecycle classifications. These are catalog/example counts. The selected system boundary and authoritative requirement inventory determine actual clause coverage, and independent assessment establishes satisfaction. Unexecuted fixtures keep empty observed-result fields. [D8]

The proposed demonstration adapts the existing reporting-worker flow into a controlled software-development task: a human admits a root, a scheduler creates distinct build and test workers, workers read approved repositories and write designated artifacts, and a protected release operation requests additional approval. The current executable still uses its reporting fixture; the DevSecOps adaptation is a proposed integration, not relabeling the test results. Proposed measurements include denied destination substitution, forbidden artifact export, restricted child minting, concurrent replay, parent fencing, and preserved outcomes after interruption. [D1, §14; D8; N2–N3]

NIST’s agent project provides an engagement route for that applied example. Formal NCCoE collaboration follows the applicable notice and agreement process. A cryptographic-module submission follows its actual module and laboratory boundary; CMVP/CAVP do not themselves test industry-protocol compliance. OLIR uses its prescribed agreement and mapping templates. The current submission is a technical contribution and review request, not a universal application or an assertion of any of these approvals. [N4–N7]

I.10. Source precedence and preserved distinctions

The exact delivered code defines executable behavior for `http6-rw-review-1.0`; its schema and tests document that same profile. The original full architecture defines W1/W2/N1 targets. The mathematical formalizations define their probability and access assumptions. The NIST mapping defines engineering relationships to selected publications. The updated IEEE paper provides the public theory, worked examples, current author credits, and cited SSL history. No source is silently used to broaden another source's theorem or implementation result.

Several distinctions particularly matter. The full architecture contemplates encrypted records and lightweight continuing authenticators; the review program uses native RW words with HMAC and visible synthetic payloads. The full policy schema supports much richer conditions than the review program's tuple rules. The original architecture presents candidate public payload-digest fields, while the later IETF candidate refines sensitive-field placement; a final native codec must resolve this choice explicitly. The later Lean registry is a source cross-reference, not a new kernel receipt. The unreadable unnumbered formalization is listed as unavailable rather than assumed identical to its numbered counterpart. [D1–D8; M4]

The source archives retain their original dates and language. Newly written submission pages identify the November 10 compilation, current routing, and new evidence. They do not retroactively change the authorship of earlier mathematical documents. Advisor credit follows the owner's supplied record; the historical SSL sources establish the historical account, not coauthor assent or approval of this submission. The release checklist records those approvals separately. [D4]

I.11. How to use the complete payload

Begin with the technical contribution in `00_submission`, then read this specification and the full architecture in Part II. Part III provides the exact mathematical and executable attachment. Appendix A contains the complete delivered envelope schema. Appendix B indexes all sixteen payload cases. Appendix C identifies the remaining schemas and machine-readable models. The full local source and instructions are in `04_validation`; the complete assurance source, appendices, examples, and independent reference code are in `05_assurance`.

The compiled review binder adds the latest IEEE theory, the payload-evidence white paper, and the 159-page expanded NIST review volume. Original source PDFs and Markdown remain independently usable. Checksums authenticate byte consistency against the supplied manifest; they do not constitute a publisher signature or a security certificate. Nothing in this packaging operation sends mail, publishes a draft, enrolls a collaborator, or commits the authors to a disclosure agreement.

I.12. Reference register for this compilation

[D1] *HTTP6: Proof-Native Web Protocol, Process-Bound Oscillation and Human-Authorized Fleet Recovery*, November 8, 2026. Full 17-section architecture; reproduced in Part II. Original Markdown and illustrated PDF retained in `01_specifications`.

[D2] *HTTP6 / Rosario–Wang — Reconstructed Validation Kit*, Kit 1.1, original core 1.0. `04_validation/HTTP6_RosarioWang_Validation_Kit/README.md`, `QUICKSTART.md`, and detailed guides.

[D3] *HTTP6: Executable Rosario–Wang Proof and Observer Demonstration*, November 9, 2026. Reproduced in Part III; original retained in `01_specifications`.

[D4] Rosario et al., *HTTP6: From Durable Credentials to Continuous, Hereditary Authority*, author and SSL-history update 1.1. Updated theory, contributor record and cited SSL history retained in `02_white_papers` and `00_submission`.

[D5] `http6_rosario_wang_demo.py`, core version `http6-rw-review-1.0`, unchanged. SHA-256: `fb5fb232e96967309ddd9439c5c902fb60876038a9fba007f84e81903f111930`.

[D6] Original envelope and fixture schemas, payload cases, and two public native vectors in the reconstructed kit; indexed in `07_schemas_and_examples`.

[D7] *Hereditary Authority from the Human Pilot Light* and `Authority_Policy_Schema.json`; full source and schema retained.

[D8] *HTTP6: Human-Originated Authority for NIST-Aligned Cybersecurity*, Expanded Public Review Edition 2.0; complete 159-page volume, modular chapters, twelve appendices, schemas, examples, models, source and prior evidence retained in `05_assurance`.

[M1] *Rosario–Wang: Consolidated Construction*, September 16, 2026. Response equations, exact verification, entropy, burn state, sequential privacy and active-experiment distinctions.

[M2] *A Unified Geometry of Private Navigation and Observable Projection*, September 13, 2026. Fibers, private slices, navigation, distributional criteria, and exposure analysis.

[M3] *Rosario–Wang: Complete Formalization of Geometric Incidence Encoding, Transcript Privacy, and Private-Reference Recovery*, numbered `(2)` attachment, September 14, 2026. Exact calibrated construction and 40-symbol family.

[M4] *Appendix B — Lean 4 Formalization Registry*. Declaration correspondence and recorded validation boundaries; no new kernel run in this compilation.

[E1] `06_evidence/regeneration_2026-11-10/validation_report.json` and accompanying logs, vectors, public records, effect databases, and benchmark. Freshly executed local reference evidence.

[N1] NIST NCCoE, *Agent Identity and Authorization project page*, rolling-feedback statement. Checked November 10, 2026. `https://www.nccoe.nist.gov/projects/software-and-si-agent-identity-and-authorization`

[N2] NIST, September 29, 2026 project bulletin; DevSecOps use case and `Ai-Identity@nist.gov`.
`https://content.govdelivery.com/accounts/USNIST/bulletins/42d04b7`

[N3] NIST NCCoE, *Summary of Comments on the Concept Paper*, standards/extensions/profiles direction.
`https://pages.nist.gov/nccoe-ai-identity/summary-of-comments.html`

[N4] NIST NCCoE, *Collaborate with Us: Technical Contributions*, formal project participation and letters of interest.
`https://www.nccoe.nist.gov/get-involved/collaborate-us-technical-contributions`

[N5] NIST, *CMVP FAQs*, protocol-versus-module boundary, SG-6.
`https://csrc.nist.gov/projects/cryptographic-module-validation-program/faqs`

[N6] NIST, *OLIR Submissions*, required agreement and mapping templates.
`https://csrc.nist.gov/projects/olir/olir-submissions`

[N7] NIST, *Crypto Reading Club*, personalized research inquiry instructions.
`https://csrc.nist.gov/projects/crypto-reading-club`

[N8] NIST NCCoE, *DevSecOps Practices project page*, currently reviewing comments; retained earlier search listing identified November 9, 2026 as its fixed-comment deadline. The rolling agent-identity route is handled separately.
`https://www.nccoe.nist.gov/projects/secure-software-development-security-and-operations-devsecops-practices`

Part II. Complete original HTTP6 architecture

The following architecture text is reproduced from the supplied November 8 specification. Heading identifiers are qualified by part for navigation. Its five original diagrams are recovered from the source PDF. Its architectural requirements and proposed identifiers retain their original scope; the executable profile is specified separately in Part I.

Citation keys within this preserved part refer to its own source register.

> **Architecture thesis.** HTTP6 pairs an enrolled Human Admission Client circuit with an authenticated Server Circuit Twin. The human establishes one live process generation; every request, response and heartbeat advances a distinct, encrypted, one-use record proof. Machine agents join by attestation and route policy. Recovery creates fresh parent and child epochs through approved machines, independent recovery TEEs or a human recovery superuser.

HTTP6 deployment	Customer/client boundary	Server placement	Channel security
W1 — compatibility	Admission web/extension workspace	Wasmer Edge HTTPS gateway, WASIX workers	Inner HTTP6 proof and encryption; outer HTTPS
W2 — managed browser	Extension view with native companion	HTTP6 gateway + protected host	Native process-state protection; selected HTTPS carrier
N1 — native HTTP6	HTTP6-aware browser or native client	Native HTTP6 listener + WASIX sandbox	HTTP6 peer handshake, authenticated encrypted records

II. Introduction and design thesis

II. Abstract

HTTP6 is the proposed next-generation web and API communication profile built on the Rosario–Wang proof construction and the human-authentication experience of the Human Admission Client. It is designed to replace the conventional separation between a long-lived transport connection and a repeatedly accepted application bearer credential with an authenticated pair of living circuit executors. A person initially proves control of an enrolled private reference through the Human Admission Client's locally rendered challenge geometry. The accepted proof authorizes exactly one client runtime generation, linked to a specified server twin, an admitted machine, an allowed route, and a declared policy. The two executors then enter **oscillation**: an ordered exchange in which each request, response, heartbeat, and selected asset block carries a fresh proof of participation in that particular channel state. The service encrypts its replies from the same authenticated channel lineage that the client uses to decrypt them; direction-separated keys, sequence values, and message-specific nonces make every record cryptographically distinct.

HTTP6 is conceived as a **full channel protocol**, not a relabeled HTTP header or an alternative name for OAuth. Its native deployment supplies peer authentication, protected setup, confidential and integrity-checked content, process-bound authorization, application-level origin policy,

bounded sandboxes, and one-use consumption throughout the transaction lifecycle. A browser user should be able to open a document, obtain verified HTML, load images and scripts, follow a link, submit a form, download a file, and conduct an ongoing conversation through the same protocol lineage. A machine agent should be able to perform corresponding operations against an enrolled server twin under machine, route, and delegation policy. HTTP6 therefore addresses the transport and authorization layers together while preserving separate responsibilities and making every security decision visible to the implementing architect.

The first deployment can be carried inside a conventional HTTPS connection to a Wasmer Edge application, so existing browsers can participate. The final **native HTTP6 channel** replaces TLS on explicitly HTTP6-aware transports, using its own specified cryptographic handshake and record protection. These profiles share the same proof and policy model; their transport packaging differs. This blueprint defines the normative target protocol, protocol state machines, API surfaces, sandbox boundary, source modules, rollout stages, and release tests. It is an architecture specification and engineering plan rather than a claim that an HTTP6-aware browser, public protocol registry, or production-safe TLS replacement has already been deployed.

II.1. The architectural question

When a customer types an address into a browser, modern clients conduct a long series of separable activities: locate the host, establish a transport, negotiate confidentiality, request a resource, receive a response, apply security policy, parse the document, discover additional resources, and repeat. A website login may occur at any point during this process. Identity and permissions subsequently flow through cookies, OAuth access tokens, session records, or signed assertions. The objects that establish permission can be replayed wherever the receiving service accepts possession or inadequate binding as sufficient.

HTTP6 changes the unit of authorization from *possession of an authentication artifact* to *live participation by a particular approved executor in a particular authenticated exchange*. The benefit is particularly clear for requests that change state: an attacker who captures the previous message has acquired a consumed observation, rather than an authority object usable for future API requests. This depends on the receiving service checking the fresh message proof and consumption record at the actual business-effect boundary.

The browser-session goal is equally important. The user should answer the original admission challenge geometry once to launch an admitted runtime. After this human admission, an active and synchronized executor pair can authorize an ongoing sequence of navigation, asset retrieval, and service messages through machine-generated per-record proofs. The person reauthenticates when a policy event terminates that particular runtime generation, such as browser shutdown, circuit termination, a changed network route, a revoked account epoch, or an explicit privileged-action requirement. The resulting model preserves a convenient continuous browsing experience **while attaching authority to each message**, rather than relying on an unexamined time interval during which a stolen cookie is sufficient.

An installed browser extension may present the first user experience inside a controlled workspace. A native companion can provide transport and process attestation outside JavaScript. A vendor can place a WASIX service behind a compatible web gateway before it deploys HTTP6-native network listeners. The protocol deliberately separates what the browser can currently delegate to extensions, what a native host must implement, and what only a browser-network-stack implementation can make first-class.

II.2. Names, products, and technology layers

The **Human Admission Client** is the customer-facing application that enrolls a person, displays the geometry challenge, obtains the person's directional response, and explains the requested origin and operation. The **HTTP6 reference architecture** is the cryptographic and operating platform (Proof Authority, Enrollment Service, Minting and Delegation Service, Policy Decision Point, Authoritative State Plane, and Consumption Ledger) that manages circuit families, custody, admission, execution sandboxes, proof checking, authoritative state, and service-to-service communication. **Rosario–Wang** identifies the proposed private-reference and proof construction used by these components. **HTTP6** names the proposed network/web protocol that transports authenticated and encrypted proof-bearing records between admitted client and server executors.

The name HTTP6 is a working protocol identifier. It is distinct from IPv6 addressing, HTTP/2, HTTP/3, and the current IETF HTTP-version namespace. Initial experiments can use an enterprise-owned scheme such as `http6://` only inside an explicitly installed and trusted resolver. The public Internet uses normal schemes until protocol registration and browser support have been designed and accepted. A proof-bearing HTTPS gateway preserves a practical deployment bridge.

A **circuit binary** is an admitted executable artifact representing a particular proof and authorization capability. A **circuit runtime** is a single executing instance of that artifact, with a measured implementation, local volatile state, a process generation, and, where supported, host-backed identity material. A **twin** is the matching remote executor responsible for one channel's state. The twin relationship is sometimes described as *entanglement*; in HTTP6 it means cryptographically synchronized state machines, not physical quantum entanglement. Each side has a matching context and independently verified role, while communication advances through directional sequence positions.

A **proof** in the opening ceremony is the person's solution to a fresh admission challenge evaluated by the authorized Proof Authority. A **record proof** is a lightweight, fresh, message-bound cryptographic authenticator generated by an admitted circuit runtime. A request and its response have separate record proofs. A **channel epoch** identifies one established twin pair; a **process generation** identifies one instance of the client process; a **route generation** identifies the normalized network path admitted by policy. Any strict route or process transition creates a new epoch and, for human policy, requires a new human proof.

II.3. Trust principles and core protocol invariants

HTTP6 has ten governing invariants. These invariants apply to the full native profile and shape the compatibility profile's application-level checks.

Invariance of server identity. The client authenticates an approved destination and its registered HTTP6 service identity before the user is invited to answer a challenge. Service availability messages are signed or otherwise protected by an enrolled trust anchor. The displayed vendor identity and origin come from that verified context, addressing deceptive cross-origin challenges and replayed advertisements.

Invariance of human admission. The geometry response is consumed for one opening ceremony bound to the exact server, channel proposal, client runtime measurement, request context, and account epoch. The process admitted after success is the same measured runtime that con-

tributed its fresh channel public material during the ceremony. A different runtime instantiates a different channel proposal and requires its own admission.

Invariance of twin identity. Client and server name both enrolled peers, roles, accepted circuit identities, and expected executable measurements. Each side confirms the complete handshake transcript. The server demonstrates its own approved executor status, rather than letting a counterfeit endpoint accept a user response and impersonate the remote twin.

Invariance of process lifetime. A human-authorized runtime is tied to a process generation. A normal shutdown closes the channel. A process crash or lost heartbeat causes the partner to retire the channel when its liveness policy expires. Relaunch creates a new generation. Human policy requires a fresh admission opening proof to create that generation; machine-agent policy can authorize a fresh attested reboot through a distinct rule.

Invariance of route generation. HTTP6's strict human profile binds the observed, verified network route to the channel epoch. Any changed source/destination endpoint, interface identity, gateway identity, or explicitly monitored IPv6 tuple suspends further application effects and starts a new admission ceremony. Network transitions therefore cannot silently carry a valid session to a substitute machine or path. IPv4, IPv6, and MAC addresses are policy attributes; cryptographic machine identity and runtime proof provide the primary origin evidence.

Invariance of per-message authority. Every accepted request and response contains a unique directional position, a canonical payload commitment, a current-channel reference, and a fresh record authenticator. The receiving twin checks the current state before releasing the content to the application. Previous receipts and copied witnesses are consumed history.

Invariance of directional confidentiality. Both twins derive compatible traffic secrets from one authenticated handshake lineage and then derive **separate keys and nonces** for client-to-server and server-to-client operations. They never encrypt two different records with the same key/nonce combination. The user describes a shared entropy pool; the implementable interpretation is a shared secret key schedule with independently mixed local randomness, domain-separated derivation, and authenticated ratcheting. Synchronization uses matching context, rather than repeated raw randomness.

Invariance of business-effect consumption. The server reserves or claims each permitted operation in durable state before executing it. The response's proof is a distinct event bound to the operation result. Ambiguous failures use same-operation reconciliation and preserve previously consumed positions, avoiding a second business effect after a retry.

Invariance of sandbox authority. WASIX/WASM executors receive only declared filesystem, network, clock, randomness, and host-call capabilities. The HTTP6 gateway keeps key custody, raw proof material, and authorization records in designated services. A compiled circuit copied from storage requires the matching admitted runtime, trusted device attributes, fresh state, and relevant human proof before participating in a protected channel.

Invariance of protocol identity. Both peers authenticate the selected HTTP6 profile, algorithms, route policy, and version. An attacker who downgrades a native connection to a weaker mode obtains a failed handshake; compatibility is enabled by an explicit vendor policy rather than by untrusted negotiation.

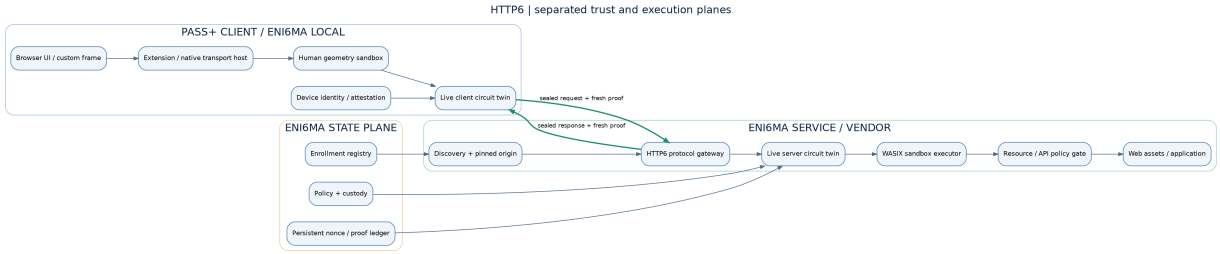

Figure 1: HTTP6 trust and execution topology, from the Human Admission Client through the Authoritative State Plane to WASIX resource services.

II.4. What changes from the conventional stack

The existing web stack uses DNS and web discovery, TCP or QUIC, TLS, HTTP request/response framing, browser origin rules, and application authorization. TLS is an essential secure-channel mechanism in today's deployments; JWT is one possible credential encoding at the application layer, while an OIDC ID token has a distinct login purpose. HTTP6's goal is to replace both *native-channel security* and *reusable bearer authority* inside a qualified HTTP6 route. This includes peer authentication, encrypted records, integrity and freshness, per-request/response proof, one-use state, and process and route lifecycle rules. A native connection should no longer depend on a JWT expiry window to authorize its next operation.

A sophisticated existing alternative, OAuth DPoP, already provides proof-of-possession binding for tokens and a unique proof per HTTP request. HTTP6 therefore should be compared against both ordinary bearer-token deployments and sender-constrained deployments. Its proposed additional design dimensions are a human-opened execution generation, admitted circuit twins, explicit process and machine identity, payload-bound bidirectional records, retained one-use state, controlled route mobility, and WASIX sandbox enforcement. A persuasive security analysis treats these as composable protocol obligations with explicit attack tests.

The resulting authorization rule can be written as a predicate:

```
ACCEPT=VerifiedPeerANDAdmittedCircuitANDLiveGenerationANDApprovedRouteANDFreshSequenceANDAuthenticRecordANDPolicyPermitsANDStateConsumed
```

A business resource receives the decrypted request only after these checks pass. An HTML renderer receives the document only after the response passes the corresponding reverse-direction checks. The same rule scales from a first page load to an API call and an agent-to-agent exchange.

II.5. The exact page-to-asset lifecycle: fourteen protocol phases

An HTTP6 implementation succeeds only if it can explain every transition between a human entering a URL and a browser displaying a verified page. The protocol is organized as a series of independently testable phases. A client can retain a long-running admitted executor while it requests multiple resources from one service twin; authorized cross-origin requests create distinct paired channels. On a new human profile, an accepted first human proof establishes the

lifetime of the client process. Subsequent assets, links, and application messages obtain individual record proofs through the active oscillation rather than asking the human to solve geometry for every image.

Phase 0: local readiness. The Human Admission Client loads the locally packaged, signature-verified client runtime. It establishes a process-generation ID, a protected memory compartment, and the device identity policy for this host. It reads trusted HTTP6 discovery roots and permitted site policies. A process launched under ordinary browser JavaScript has the assurance offered by that runtime; the higher-grade profile delegates channel state to an attested native component or suitable protected execution environment.

Phase 1: URL interpretation. The customer enters an HTTP6 resource name in the trusted HTTP6 workspace. The resolver parses its scheme, canonical authority, port, resource path, and requested operation. It rejects ambiguous encodings and checks local policy. In an ordinary Chrome installation, the address is entered in an extension-owned page or an approved native handler; an arbitrary third-party website does not thereby gain the ability to instantiate an HTTP6 peer.

Phase 2: authenticated service discovery. The client resolves a registered HTTP6 service descriptor using a pinned registry entry, verified DNS-backed service record plus independent trust, or a configured enterprise catalog. An IP address and DNS response are routing information. The descriptor binds an enrolled service identity, key or attestation trust root, supported protocol versions, policy, approved WASIX service release, discovery expiry, and supported transport profiles. The client verifies the descriptor before presenting the named service to a human.

Phase 3: availability and server proof. The server returns a fresh `SERVER_HELLO` associated with a signed or attested service identity. It contributes a fresh ephemeral public key, a nonce, an executable measurement, the accepted route rule, and its challenge of the client's proposed generation. This is the server's one-shot availability proof. The client verifies it against the discovered server identity; a copied advertisement from a different service or an unrelated process fails the expected transcript binding.

Phase 4: client circuit proposal. The client starts a new measured circuit process, creates an ephemeral secret and public value, and commits to the requested origin, intended account, client process generation, proposed route, and server hello. A TEE-backed profile attaches current device and runtime attestation. The two sides compute a canonical handshake transcript hash from both identities, offers, challenges, and agreement values. This transcript is the security context of the upcoming human proof.

Phase 5: human challenge projection. The server chooses the admitted Enrollment Service record and generates the fresh challenge for the registered circuit. The locally running circuit projects the current geometry in a trusted Admission Presenter screen. The person sees the verified server identity, scope of requested access, and current geometry; the private knowledge and mapping are used locally to form the response. The client sends a protected, transcript-bound response envelope through the participating process.

Phase 6: admission decision and one-shot consumption. The Proof Authority checks enrollment, registered circuit, selected grade, origin, current account epoch, challenger identity, and proof response. It claims the challenge before execution and commits an accepted or rejected result in durable state. Human admission belongs to the exact client process generation and

server twin committed in phases 3 and 4. A previously recorded geometry solution cannot be used to admit a different process or server context under these validation rules.

Phase 7: protected channel key schedule. The client and server mix authenticated ephemeral key agreement, fresh nonces, and the complete handshake transcript into a domain-separated key schedule. Both derive compatible traffic secrets for their roles and an independent exporter for record-proof generation. The proof establishes authorization; cryptographic key establishment supplies channel confidentiality. Either side confirms the same transcript with a keyed `FINISHED` record before application data is accepted. The server records a unique channel epoch and a policy-governed peer route generation.

Phase 8: first resource request. The browser asks for the original document (`GET /`). The active client circuit creates one `REQUEST` record, carrying the declared method, normalized route and resource ID, content preferences, operation ID, message sequence, predecessor commitment, and payload digest. It encrypts and authenticates the record, then supplies the HTTP6 proof bound to those exact bytes. The server twin verifies and consumes the message position and calls its policy enforcement point.

Phase 9: sandboxed service execution. After validation, the HTTP6 gateway invokes an admitted WASIX/WASM service with restricted capability handles. A web handler returns HTML or a manifest; an API handler performs the requested operation. Stateful effects use a durable claim and idempotency register. The application keeps its own data authorizations, and the sandbox obtains only the network and filesystem capabilities required to produce the response.

Phase 10: server proof and encrypted response. The service twin creates a distinct `RESPONSE` record. It binds the original request ID, response status, declared MIME type, content digest, sequence position, and the appropriate integrity metadata. The server encrypts the payload with a response-direction key derived from the shared channel lineage, attaches a fresh response proof, and commits its outbound state. The client verifies this distinct record before exposing bytes to its renderer.

Phase 11: page construction and asset discovery. The trusted renderer parses an admitted content type and applies the HTTP6 origin and content policy. HTML refers to CSS, fonts, media, scripts, or image objects. Each fetch is represented as its own `GET`, or as a stream of individually authenticated asset chunks. A content manifest can bind a tree of resource hashes for faster integrity checking, while every transmitted proof still advances the appropriate channel state. External origins require their own authenticated service identities and allowed channels.

Phase 12: link navigation or application event. When the person clicks a link, submits a form, or requests an API operation, the client derives an operation from the actual activated element and current document context. It canonicalizes the destination and payload, checks permission to follow that link, and advances a fresh proof on the relevant twin channel. Cross-custodian links open a separate paired channel, subject to new human approval only where the local delegation and site policy require it.

Phase 13: heartbeat, synchronization, and shutdown. Idle channels exchange bounded `HEARTBEAT` or `ACK` records, each with a unique proof, sequence, and liveness commitment. On normal exit both twins close the epoch. On missed heartbeat the server freezes new business effects until the peer completes the configured continuity policy. A changed IP, IPv6 address, network route, process, or sandbox identity in a strict human profile starts a new admission generation; recovery is a fresh handshake, never an old ticket silently carried to the new path.

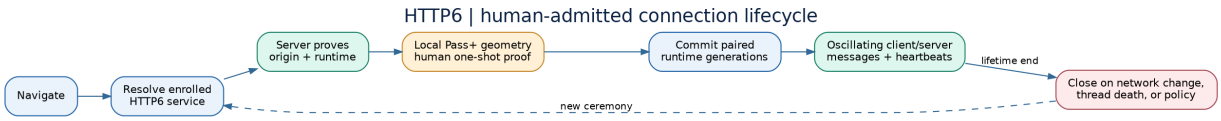

Figure 2: HTTP6 end-to-end request lifecycle, from discovery through proof and runtime-scoped communication.

II.6. The opening human proof and synchronized twin oscillation

The initial admission ceremony is a **root admission**, whereas subsequent request and response proofs are **runtime-generated continuations**. This difference makes the protocol convenient for browsing. The user establishes who may operate the local circuit and what server may pair with it; the active circuit then generates per-message cryptographic evidence as the paired state machine advances. A human need not individually answer each image or every kilobyte of a streamed video; the policy may nevertheless demand a new interactive proof for a payment, credential export, administrative change, or other high-impact action.

Oscillation is a two-party state transition. Suppose the live client is at `(epoche,C2Ssequence41)` and the server is at `(epoche,S2Csequence17)`. A client request advances only the client-to-server direction, consumes its position, and records the request commitment. The server's response advances the server-to-client direction and explicitly refers to the request. Both maintain an authenticated predecessor chain. Simultaneous messages use distinct directional chains with bounded in-flight windows; serializing the entire system across all channels is unnecessary. Record-delivery acknowledgement commits remote observation, while application-result acknowledgement confirms the business effect. These are different events.

The term *shared entropy pool* is formalized as an **authenticated common secret lineage**. Entropy originates at both participating peers during handshake and, where specified, during rekey. Each peer has the information needed to derive the same sequence of per-direction traffic keys and proof material. The peers never simply transmit a static pool of secret random bytes and repeatedly encrypt under identical nonce/key values. This approach yields synchronized cryptographic state while maintaining distinct record secrets. A server twin can operate inside a bounded WASIX sandbox while a dedicated custody/attestation service supplies its authorized key schedule and message operations.

An on-path adversary can relay bytes, so the human geometry challenge alone does not authenticate the remote origin. HTTP6 couples that human admission with independently authenticated server identity, a transcript-bound ephemeral exchange, client process identity, role-specific proofs, and final key confirmation. A transparent relay then delivers traffic to the legitimate endpoints without gaining the key material or creating a second admitted runtime; a counterfeit endpoint fails peer and transcript confirmation. A compromised authorized endpoint requires separate endpoint-hardening and resource-policy controls. These distinctions keep the anti-interception design aligned with its actual trust mechanisms.

II.7. Session lifetime and route transitions

HTTP6 treats the process generation as a live security principal. A service can mark an HTTP6 channel `ACTIVE`, `SUSPECT`, `DRAINING`, `CLOSED`, or `REVOKED`. The client and server separately maintain a last-accepted proof, generation identifier, and liveness deadline. User-visible status reflects

these states. The initial admission may open several destination-specific twins, each with its own channel epoch, while a parent policy records their common human origin.

The strict mobility policy serves applications that prefer security over seamless network switching. If the observed network tuple or required egress path changes, new protected operations pause; the runtime may inform the customer and close the previous epoch. Returning to the service from a different Wi-Fi network or a new IPv6 prefix therefore triggers a fresh human ceremony. A mobile profile can instead permit reattachment only through a separate attested rebind proof and policy; this version of the specification selects **strict re-admission for human browser sessions**. Neither source IP nor MAC alone proves the physical machine's identity; an encrypted and attested runtime supplies the primary continuity check.

On the server, a reboot or worker replacement also changes generation identity. A previous process cannot reappear merely by replaying a persistent record. A trusted server fleet can create a new twin through authenticated deployment policy. In the ordinary human-authored profile, an unexpected client restart requires a fresh admission proof. For an autonomous service, the restart decision can be delegated to attested machine policy or explicitly routed to an operator.

II.8. HTTP6 protocol model, roles, and records

The first specification edition uses the working identifiers `http6.w1` for the proof-bearing HTTPS-compatible profile and `http6.n1` for the native protocol. The names are project-defined version strings. They are not registered Internet standards. A wire implementation requires a published binary codec, test vectors, and deterministic serialization. The initial source model below is therefore an implementation contract rather than a claim of an existing interoperable endpoint.

There are seven protocol roles. The **Admission Presenter** displays the verified requesting service and the local challenge. The **Client Circuit Executor** is the measured, process-bound runtime. The **Service Discovery Authority** vouches for destination descriptors. The **Server Circuit Twin** participates in the handshake and record exchange. The **Proof Authority** evaluates initial human admission, registered agents, and retained current state. The **WASIX Application Sandbox** executes admitted web/API handlers through bounded host capabilities. The **Policy Enforcement Point** gates external data and business effects before the application receives a request. A single host may combine several roles under different credentials; the security contract names their independent decisions.

A channel begins in `NEW`, moves through `DISCOVERED`, `SERVER_ATTESTED`, `CLIENT_PROPOSED`, `HUMAN_PENDING`, `ADMITTED`, `KEY_CONFIRMED`, and `ACTIVE`. A failure moves it to `REJECTED` or `HELD`. A replay or retired epoch moves it to `REVOKED`. These states are durable enough that a process crash cannot turn a previously consumed challenge into a fresh one. A server operation has its own substate: `RESERVED`, `CLAIMED`, `VERIFIED`, `DISPATCHED`, `COMPLETED`, or `UNCERTAIN` with terminal failure categories. A server cannot infer a completed business effect solely from an authenticated transport ACK.

II.8.1 Normative envelope fields

Every protected application record carries one immutable, canonical header and one cryptographic body. The prototype schema uses deterministic CBOR or another strictly specified binary

encoding; JSON below is for readability. The sender includes an explicit record class (`REQUEST`, `RESPONSE`, `ASSET_CHUNK`, `HEARTBEAT`, `ACK`, `CLOSE`, or `REKEY`), a protocol version, channel and epoch identifiers, peer role, direction, sender process generation, normalized route ID, sequence, predecessor digest, operation ID, optional referenced request, policy epoch, content type, expiry/hold rules, and a digest of its canonical payload. The protected body consists of the authenticated ciphertext, AEAD nonce or derived nonce reference, and a fresh proof authenticator.

```
{
  "protocol": "http6.n1",
  "kind": "REQUEST",
  "channel_id": "opaque-channel-id",
  "epoch": 8,
  "direction": "client_to_server",
  "sender_generation": "opaque-runtime-generation",
  "sequence": 42,
  "previous_commitment": "hash-of-prior-directional-record",
  "operation_id": "unique-operation-id",
  "route_id": "admitted-route-identifier",
  "method": "GET",
  "resource": "/documents/report",
  "payload_digest": "hash-of-canonical-plaintext",
  "policy_epoch": 3,
  "ciphertext": "AEAD-protected bytes",
  "record_proof": "fresh-directional-authenticator"
}
```

A parser rejects duplicate fields, unknown critical extensions, out-of-range lengths, reinterpreted Unicode, ambiguous path separators, and noncanonical encodings. Protected content remains opaque to intermediaries, subject to deliberate public routing metadata. Compression, streaming, and partial reads have versioned definitions; compressing secret-dependent content alongside attacker-controlled content can create an information side channel and therefore requires its own reviewed profile.

II.8.2 Cryptographic key schedule

Initial authenticated key establishment derives a transcript secret from the approved ephemeral agreement. The service identity is independently authenticated and the client's attested ephemeral key is bound to the same transcript. A possible conventional composition uses a well-reviewed KEM/ECDH construction, HKDF, an approved AEAD cipher, and a separate MAC construction. HPKE (RFC 9180) and TLS 1.3 (RFC 9846) provide relevant engineering models for key establishment and authenticated channels; HTTP6's native use requires a separate exact composition rather than assuming these standards supply an HTTP6 wire format.

The shared lineage can be modeled as `Kroot=KDF(ephemeral_secret,transcript_hash,fresh_context,enrolled_identity_binding)`. Separate labels derive `Kc2s_encrypt`, `Ks2c_encrypt`, `Kc2s_proof`, `Ks2c_proof`, and distinct rekey materials. Each record uses a unique `(epoch,direction,sequence)` derivation; the resulting AEAD nonce remains unique for every invocation of the selected cipher. The per-record proof authenticates the canonical header, ciphertext digest, operation commitment,

previous accepted record, and live runtime generation. A proof accepted at position 42 is different from position 43, even for repeated plaintext, and a server proof cannot substitute for a client proof because its direction-specific keys differ.

The **entropy pool** is defined as coordinated, cryptographically derived state, rather than two processors reading identical exported random bytes. The clients' fresh contributions are mixed into the handshake and the derived secret schedule. Regular rekey uses the authenticated prior ratchet and a fresh contribution according to a specified hybrid derivation. The KDF can be deterministic because its secret inputs have sufficient independent entropy; proof codes, a human-entered secret string, a timestamp, and a public challenge are excluded as the sole inputs to encryption strength. Historical keys are erased from active memory according to the forward-secrecy and retention policy.

A design asserting unconditional or information-theoretic transcript privacy must separately prove that the exact observable record—including headers, sizes, timing, failures, and rekeys—satisfies its stated distribution assumptions. The implementation here uses concrete authenticated cryptography for channel protection and retains the private-reference construction for human admission. This makes protocol roles precise without conflating a transcript theorem with all properties of a deployed web transport.

II.8.3 Request and response proofs

For a client request, the policy engine authorizes the tuple `(owner,service,runtime,route,method,resource,canonical-payload-digest,sequence,epoch)`. The server verifies both ciphertext integrity and the proof-circuit record authenticator under the named current state before invoking a handler. For a server response, the client verifies `(channel,referenced_request,server_generation,response_code,response-digest,sequence,epoch)`. This creates symmetric authenticated participation: every direction proves that its sending twin controls the state of that direction.

For large media, a stream has a named parent operation and a bounded stream policy. Every chunk carries its own authenticator and sequence; video frames may map to multiple protected chunks, and a conference may have one committed track manifest plus per-packet protection. The cryptographic record proof can be a short MAC tag because the strong human proof authorized the parent executor, while the continuously advancing key schedule now proves active machine participation. The design does not require replaying the full human geometry per frame. Bandwidth and CPU budgets determine the selected record size and whether selected packets are encrypted individually or grouped into authenticated segments.

An API with write effects adds a persistent operation-consumption register. If a response is lost after a payment succeeds, the client queries the status of the **same** operation through a fresh authenticated record. The server returns the previously committed result and prevents a duplicate charge. This is the distinction between at-most-once proof acceptance and exactly-once business effect: the latter is achieved through idempotency and application reconciliation, not solely through a cryptographic MAC.

II.8.4 State, identity and routing

An HTTP6 route descriptor defines the authenticated destination service, protocol version, port and network transport, canonical origin, allowed path templates, approved destination iden-

tities, permitted IP/subnet policy, and the runtime's machine identity constraints. Remote machine identification uses attestation or hardware-backed keys. A MAC address ordinarily resides at a local link layer and is absent from routable Internet packets; it is a useful enrollment policy attribute when a trusted local agent reports it, rather than a remote cryptographic identity. An IPv6 address may identify the observed connection and can be used as a strict mobility condition, while cryptographic service identity remains the protection against masquerading.

A strict route change triggers `HOLD_ROUTE`, closes the previous channel after verifying the live peer state, and issues a new handshake and admission as required. For user-facing sessions, the selected policy mandates human reentry on process or network transition. For machines, an attested restart policy can authorize a new channel epoch after the service confirms the registered hardware, application measurement, and approved route. Neither policy reuses a previously recorded proof position.

II.8.5 Threat-by-threat security contract

Packet interception: The intercepted record contains a spent sequence position and encrypted content. A different sender lacks the matching session secrets and runtime identity; the resource checks current consumption before processing.

Executable/circuit theft: A copied WASM binary is only an artifact. Protected opening uses the machine and process admission record, current circuit epoch, custody policy and fresh proof context. Copying the program to another host produces a different runtime generation whose attempt to participate is rejected by the paired service.

Live man-in-the-middle: A relay may forward ciphertext, but channel identity, transcript confirmation, and authenticated ephemeral establishment prevent it from silently creating an independent authenticated twin. In-band phishing and a compromised legitimate endpoint remain addressed by verified destination presentation, endpoint isolation, and operation-specific approval policy.

API key theft: A legacy API key may initially remain behind a proof-gated capsule. The eventual native HTTP6 service authorizes operations through the current record proof and local policy, removing the need for a reusable bearer key at that selected resource.

Service impersonation: The server's identity and attested binary are verified before the person responds. An unauthenticated availability answer cannot select the policy or redirect the proof to a counterfeit recipient.

Reboots, lost networks, and stale backup: A retired generation and consumed nonces stay recorded in independent retained state. New process and route generations enter an admission flow, even when the same computer returns to the network. Operators can reinstate an approved fleet through the disaster recovery root procedure defined below.

II.9. Client/server APIs, headers, and errors

In the compatible transport profile, the outer HTTPS endpoint accepts a protocol-versioned opaque envelope, such as `POST/http6/v1/records` or a bidirectional WebSocket where supported. These are **proposed interface examples**, with exact names fixed by the implementing service and published SDK. The outer request performs no application action by itself; its protected

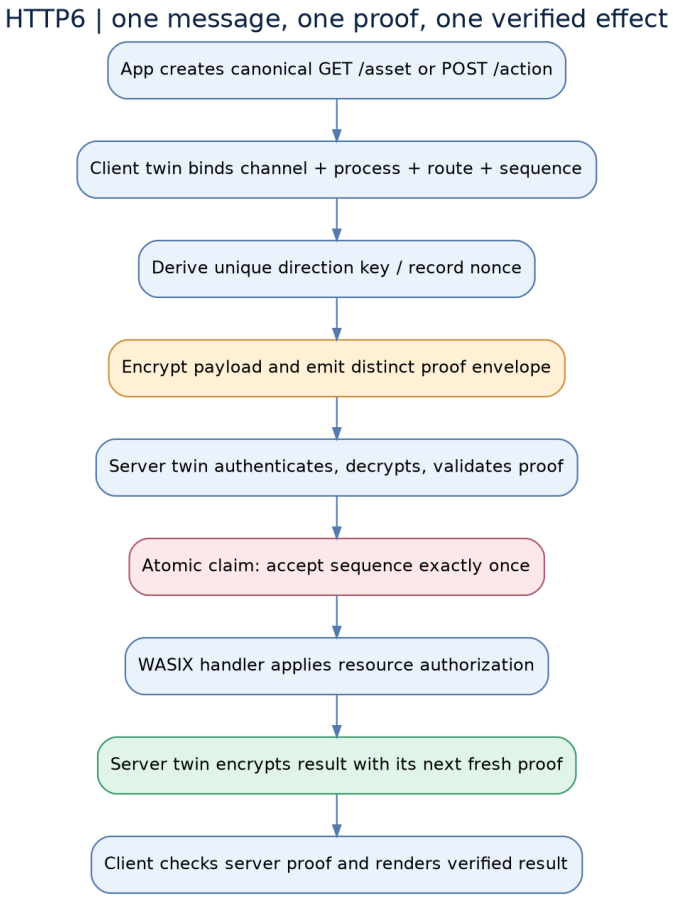

Figure 3: The HTTP6 record exchange: separate directional proofs and synchronized encryption state.

HTTP6 payload is parsed and authorized by the twin service. Ordinary HTTP status codes can communicate transport failure while the inner HTTP6 response carries the business result and proof.

In the native profile, `http6://service.example/path` selects an HTTP6-native client/transport. A future custom browser could handle the scheme and its native secure channel directly. An installed companion can register a local scheme handler and render through a dedicated browser workspace, subject to the platform's scheme registration and security model. The encrypted record framing is transport-agnostic at the application layer: reliable byte stream, QUIC-like datagrams, or a constrained message bus may carry its envelopes after the security handshake is established.

Recommended error families are `PEER_UNTRUSTED`, `ATTESTATION_FAILED`, `HUMAN_PROOF_REQUIRED`, `RUNTIME_GENERATION_CHANGED`, `ROUTE_CHANGED`, `EPOCH_RETIRED`, `SEQUENCE_CONSUMED`, `AUTH_TAG_INVALID`, `POLICY_DENIED`, `STATE_UNAVAILABLE`, `REKEY_REQUIRED`, and `OPERATION_UNCERTAIN`. Errors are authenticated and minimized so a remote unauthenticated caller cannot probe private enrollment data. Structured audit events record the allowed state transition while raw proof responses and custody keys remain within their designated execution service.

II.10. The HTTP6 browser and WASIX deployment architecture

HTTP6 needs a path from a human's normal browsing habits to a controlled execution environment. Its client strategy has three stages, each with a sharply defined capability boundary. The same protected message format and proof rules operate across the stages, so an early vendor can build an HTTP6 service without waiting for the entire public browser ecosystem to implement a new transport. The user-facing Human Admission Client retains the visible identity and proof experience; the server side of the HTTP6 reference architecture hosts the cryptographic channel, the Policy Decision Point, the sandbox, the Minting and Delegation Service, and the Consumption Ledger.

II.10.1 Stage one: HTTPS-compatible HTTP6 gateway

The first service deployment is the most attainable: an existing website or API operator deploys a new Wasmer-hosted HTTP6 gateway that uses a standard HTTPS listener. The browser opens the admission workspace using HTTPS, locally packages its circuit runtime, completes the accepted admission ceremony, then sends and receives encrypted HTTP6 frames to the gateway. Every inner request and response contains a fresh record proof. The outer TLS endpoint remains the compatibility transport for existing browsers and Wasmer Edge. The site can migrate high-value APIs to the proof-enforcing gateway before adopting a native HTTP6 transport.

The gateway handles discovery, protocol negotiation, rate limits, request-envelope validation, and routing to protected execution. The paired server twin runs under a selected WASIX/WASM runtime with bounded capabilities, and the proof authority/ledger sits in separately authenticated services. The vendor may keep existing content and database backends behind that gateway. Its only application-facing change is to expose a well-defined resource adapter that consumes a verified HTTP6 operation context.

The gateway must verify the inner proof before accessing a vendor backend. A request body copied from an earlier HTTPS exchange, even if byte-identical, is bound to a spent HTTP6 sequence and cannot trigger another effect. Existing login sessions can coexist during migration; the protected route enforces proof-first authorization regardless of a cookie's presence.

II.10.2 Stage two: extension-managed HTTP6 workspace

A Chrome extension can create an HTTP6 **workspace page** with the Human Admission Client visual identity and managed navigation controls. The user enters an HTTP6 resource address into the workspace. The extension mediates all navigation and resource fetching for that workspace through its controlled channel implementation. A remote HTML document can be displayed using a restricted renderer, a sandboxed view, or a trusted component model. URLs within the controlled view are parsed and rewritten into new HTTP6 records rather than becoming unrestricted network accesses from remote page scripts.

The extension's declared network-request API can block, redirect, or modify certain request and response headers. Its rules alone do not amount to replacement of the browser's TLS stack, nor do they provide complete control of arbitrary browser-fetched content. Chrome's `declarativeNetRequest` applies at specific browser network lifecycle stages; sensitive request bodies, response

bodies, and cross-origin page execution need other controlled components. A production extension therefore uses an explicitly trusted HTTP6 view, a packaged runtime, and optionally a native companion for its proof state and transport. A content script running in a merchant page is an untrusted input surface, with a validated bridge to the extension service worker and protected host.

A packaged extension must obey applicable installation and code-loading policies. Manifest V3 disallows running remotely hosted executable JavaScript and WASM in the extension's own execution contexts. Remote website content is treated as data or rendered in a separately specified, properly isolated origin; downloaded executable circuit material follows its permitted distribution profile. The architecture can support managed enterprise builds where selected capabilities require separate permissions, release governance, or a native component. Every renderer and compiled artifact has a signed release identity.

The extension's image, font, stylesheet, media, and document requests are all issued by the HTTP6 client and checked under the active twin. A user click is captured as an application action, not reinterpreted from arbitrary cross-origin script. An `http6://` link remains within the HTTP6 workspace unless the user explicitly selects an external web destination, in which case the appropriate normal browser security and origin rules apply.

II.10.3 Stage three: native browser transport or companion

A first-class native HTTP6 client opens its own network transport to an HTTP6-aware service, performs the enrolled handshake, and renders the verified response. It may be a custom browser, a managed browser fork, a native operating-system helper with an approved browser integration, or a dedicated desktop application. A Chrome extension can communicate with a registered native host through native messaging, subject to explicit manifest permission and allowed extension identity. Such a host can handle protected process state and transport without asking ordinary JavaScript to become a general-purpose network stack.

The native channel replaces TLS **for that connection** by implementing the same engineering responsibilities: authenticated peers, encryption, integrity, replay protection, secure key update, downgrade resistance, and error handling. Each request and response additionally carries the HTTP6 per-message proof and business authorization. A reverse proxy can simultaneously serve normal HTTPS and native HTTP6 to different client populations. Public web links, third-party APIs, package downloads, and provider integrations retain their defined transports until they adopt the new endpoint.

An operating-system network interceptor that silently rewrites arbitrary browser traffic would introduce origin confusion, protocol compatibility problems, and a large trusted code base. The preferred initial implementation is a transparent user-selected workspace with explicit enrollment, origin displays, and clear navigation semantics. A fully native browser can later give HTTP6 the same first-class URL handling that ordinary HTTP and HTTPS enjoy.

II.11. The Wasmer/WASIX service stack

The HTTP6 server is a small **protocol-facing service**, paired with a stateful **proof and authorization plane**. An organization deploys its HTTP6-capable endpoint, registers its identity and policy, and supplies handlers that accept approved operations. The original web stack can remain behind this layer. On Wasmer Edge, an ordinary app package supplies an HTTP server

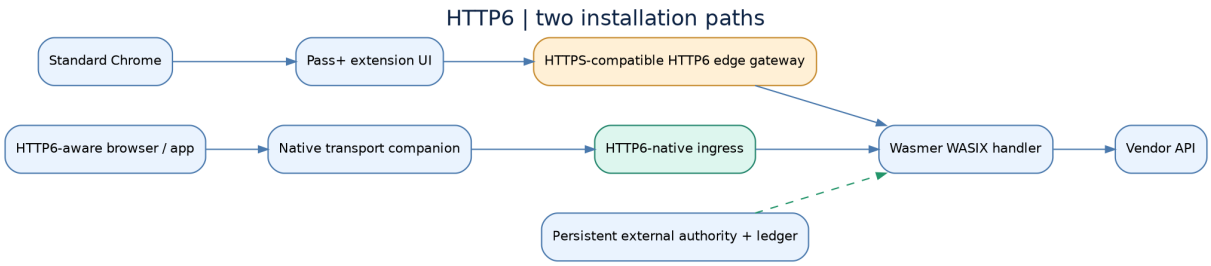

Figure 4: Deploying HTTP6 through an HTTPS gateway, a trusted extension workspace, or a native transport.

through Wasmer's deployment model. Its instances are ephemeral and may be replaced, so channel state, consumed proof positions, account epochs, business idempotency, and recovery journals must be externalized into appropriately durable storage. A process restart must be visible as a new server-generation or controlled backend reconnection, rather than quietly reusing a lost in-memory proof chain.

The service package has six primary components. The **Transport Gateway** accepts compatible HTTPS envelopes and, in a separately qualified installation, native HTTP6 records. The **Discovery Registrar** signs or references admitted service identity, route profiles, and currently approved release identities. The **Server Circuit Twin Runtime** runs only admitted binaries and emits/consumes channel proofs. The **Policy Engine** checks account, machine, route, operation, and customer entitlements. The **Custody and Attestation Adapter** accesses protected recipient material, validates machine evidence, and supplies allowed cryptographic operations. The **State and Reconciliation Store** commits sequence positions, human admission, fleet epochs, workflow results, and disaster-recovery lineage.

WASIX provides the extended WASI runtime environment and host integration appropriate to an application needing operations beyond pure computation. The deployment must select which filesystem, clock, randomness, network, process/thread, and host-call capabilities each guest can use. A sandbox is an enforcement boundary only when the host restricts its imports and protects secrets outside the guest's readable memory. Separate workers and resource budgets constrain malformed inputs and executable behavior; syscall-level filters and operating-system confinement supplement the WebAssembly boundary where deployment requires them.

Wasmer's HTTP app runtime and a native socket-listening service have different capabilities. Current Wasmer Edge documentation describes HTTP-server based app deployment and stateless, ephemeral instances, with generic TCP/UDP server support described as a future expansion. The first hosted HTTP6 product therefore uses HTTP over the platform's ingress, while native network transport can run in a separately deployed WASIX/native service or an independently managed server. This avoids making an unsupported platform feature a prerequisite for the pilot.

II.11.1 One URL, one request, one sandboxed execution

A vendor's `GET/reports/latest` endpoint registers an HTTP6 resource handler. The client performs the opening ceremony, then sends a sealed request with its message proof. The gateway

validates framing, uses the channel's admitted twin to verify the record, and asks the policy engine if this user/runtime/route can read that resource. It invokes the appropriate WASIX handler with a temporary read-only data capability and a bounded output budget. The handler returns an HTML document and a signed or committed asset manifest. The server twin protects the response, and the client releases only verified bytes to the renderer. Every subsequent asset fetch follows the same path.

For state-changing requests the gateway calls the business effect controller only after the operation is atomically claimed. This permits a conventional database or payment service to remain in place while the HTTP6 gateway replaces the reusable credential as the initiating authority. Retrying a lost response returns the outcome of the same operation rather than replaying the effect. Resource ownership and transaction limits are still checked by the consuming application.

II.11.2 Media and large streaming flows

Interactive media uses sustained twin channels. At channel creation, the participant authorizes a conference, stream role, track policy, and maximum resource budget. A centralized conferencing server can distribute a track to multiple enrolled destinations through independent HTTP6 edges, with each recipient's encrypted frame proof tied to its own channel. A multiparty mesh creates individual pairwise channels or a group state protocol with membership epochs and sender-specific keys. HTTP6's first specification selects pairwise delivery for simplicity and defines a separate group-key management profile for high-scale mesh conferencing.

Each protected media record carries a unique authenticator, sequence number, and stream/track identity. Key ratchets can advance every chunk or at fixed boundaries with a record-unique derivation. Application latency, jitter buffering, retransmission and packet loss handling must be measured: a sender that loses a packet must not create an uncontrolled proof replay; the receiver can detect missing positions and request authenticated recovery. The conference server is itself an admitted service with rights to forward only its specified objects. A shared public circuit attestation can establish membership eligibility, while each active channel keeps its distinct private proof keys.

II.12. Suggested source modules and repository boundaries

The architecture's source layout is proposed as a set of independently buildable, versioned packages. `http6-protocol` defines canonical binary envelopes, version negotiation, transcript hashing, error classes and test vectors. `http6-crypto` implements the selected standard KEM, HKDF, AEAD and record MAC schedule behind a reviewed interface. `http6-discovery` validates registered service descriptors and trust roots. `http6-client-core` implements the channel state machine, process generation and ProofCheck integration. `http6-admission-ui` hosts human geometry display, origin verification and session status. `http6-extension-workspace` handles managed view, link navigation, resource fetching and permissions. `http6-native-host` supplies protected process state and transport adapters. `http6-wasix-gateway` implements ingress and proof-enforcing handlers. `http6-runtime-twin` hosts paired circuit execution with bounded imports. `http6-policy` evaluates human, machine, route and operation rules. `http6-ledger` stores consumed positions and business idempotency. `http6-fleet-recovery` manages parent/child epochs, recovery TEE identities, human-superuser ceremonies and bulk restart orchestration. `http6-test-harness` contains loopback and fault-injection acceptance suites.

A release manifest binds each package to an exact source commit, compiled artifact digest, policy schema, capability import list, and test receipt. The browser and server runtimes can implement the same logical protocol in different languages while sharing byte-level message fixtures. Source changes to the proof engine, guest-host interface, cryptographic key schedule, or migration schema require an explicit version and interoperability test. Credentials for publication, authority, and runtime secrets belong to their designated host services rather than public source packages.

II.13. Human and automated authorization families

HTTP6 separates a **human admission circuit** from an **autonomous service circuit** while preserving one trust model. The human circuit opens a verified admission challenge, admits a particular local process, and establishes a scoped parent authorization. The machine circuit enters the corresponding oscillating relationship using enrolled hardware evidence, measured execution and a delegated policy. A fleet can include thousands of child circuits, but each child possesses a distinct identity, route, operation scope, and live proof state. One successful human superuser ceremony can approve a policy tree for those children; it does not produce a reusable master bearer credential that any child can copy and wield elsewhere.

A parent policy declares the maximum number of descendants, allowed service classes, approved machine inventory, process measurements, origins and ports, network route policies, permissions, transaction budgets, lifetimes, and human reapproval triggers. Child generation is a controlled authority operation. The newly authorized scheduler requests a child identity from the Minting and Delegation Service, the target machine presents its attestation, the Minting and Delegation Service checks the child's requested policy against the parent, and the scheduler issues a one-shot bootstrap invitation addressed to that exact child runtime. The child constructs a fresh HTTP6 twin with the remote service. A child cannot mint a more privileged sibling from its own local policy unless the parent explicitly delegates that operation.

This produces a hierarchy of **root human admission** $\rightarrow$ **fleet authorization** $\rightarrow$ **machine-specific child issuance** $\rightarrow$ **process-generation admission** $\rightarrow$ **channel-specific oscillation** $\rightarrow$ **per-message proof**. The human is the authority for the selected task set. Individual automated agents become authorized participants rather than containers for a common long-lived API secret. Every child can be suspended or retired by advancing its policy or fleet epoch. A fleet-wide emergency shutdown retires the parent and all descendant admission generations according to the enrolled revocation policy.

II.13.1 Machine attributes and the qualified path

An enrolled automated machine has a verified device key, attestation reference values, an approved runtime measurement, a current boot counter or equivalent monotonic identity, and an expected workload. The policy can also record network IP ranges, named IPv6 destinations, interface MAC identifiers, gateway identities, selected VLAN, data center zone, and route templates. These network features enforce placement and routing rules; the machine's attestation and private keys prove its stronger identity. The remote server checks current network observations and service-origin authentication before accepting each new channel epoch.

A machine reboot creates a fresh process generation and fresh channel keys. Under `attested_auto` policy the machine can rejoin automatically if the registered hardware and runtime pass attestation, its route satisfies policy, its fleet epoch is current, and the authority grants a new bootstrap. Under `human_on_reboot`, the machine sends a request to an administrator enrolled in the Human Admission Client and remains held until an authenticated approval creates the new generation. Under `break_glass_only`, even a properly attested machine waits for the designated emergency operator ceremony. These policies let low-risk infrastructure resume quickly while giving high-impact financial, custody, and administrative services direct human control.

II.13.2 Recovery authority tiers

The architecture uses three distinct recovery tiers, each creating fresh authority rather than reviving old sequence positions.

Tier A — ordinary attested reboot. A machine loses power, a cloud instance is replaced, or a WASIX guest exits. Its enrolled device and orchestrator supply fresh attestation; the Minting and Delegation Service verifies code measurement, route and fleet epoch; the service creates a new channel and starts its oscillation. Current event and consumption state come from durable stores. The procedure restores approved automation with minimal operator intervention.

Tier B — independent recovery TEE. An organization installs a physically and operationally separate recovery machine with hardware-backed keys and pre-enrolled, narrowly scoped disaster authority. The machine might be a solar-powered laptop with independent battery storage and an alternate network path. Its **power autonomy** and **communications autonomy** reduce common-cause failures affecting the primary data center. The TEE/HSM or equivalent protected runtime stores a designated recovery capability and monotonic state; solar power by itself supplies availability rather than cryptographic trust.

On a primary-site outage, the recovery machine attests its own identity and current code, reads the latest independently retained checkpoint or signed recovery policy, and requests a new disaster epoch. It enumerates the permitted machines and services, applies the authorized restart policy, and mints **fresh, machine-bound child bootstrap ceremonies** for the replacement fleet. If the alternate network is available, it coordinates new HTTP6 twins directly with reachable services. If all outside network paths are down, it can prepare sealed, signed recovery plans and local authorizations for later delivery to pre-enrolled recipients, with a defined validity window and monotonic epoch. Actual remote services become active only after they can authenticate the recovery authority and reconcile the currently accepted state. Offline work therefore preserves a controlled recovery plan rather than pretending to complete an unreachable remote authorization.

Tier C — human-superuser restoration. If the independent recovery TEE is unavailable, a designated administrator performs an enhanced admission ceremony using a pre-enrolled disaster-recovery route. The recovery policy may require two authorized humans, a threshold of protected shares, a sealed offline recovery kit, or a physical console at the surviving trust root for high-impact installations. The operator authenticates through locally verified geometry against a known recovery authority and approves a proposed disaster epoch, destination set, and bounded child-family policy. An accountable recovery signer commits the transition and distributes fresh authorization seeds or recipient-addressed bootstrap records to the surviving machines. This is **one human proof that authorizes a school of controlled progeny**, followed by individual machine and route admission for each child.

When a central authority and every enrolled recovery appliance have been destroyed, the installation follows its **root-of-trust reconstitution procedure**. The organization must possess a surviving pre-registered recovery trust anchor, offline protected credential material, or an explicitly governed replacement-root ceremony. If all trust-anchor material has been lost, a new root identity must be established and distributed through authenticated out-of-band administration before other machines can trust it. Human knowledge alone cannot make a destroyed signing key reappear. A sound disaster plan defines this material before deployment.

II.13.3 Disaster recovery invariants

An emergency restart **increments a recovery epoch**, independently of ordinary channel counters. Every old runtime generation is retired or suspended, and no old nonce becomes available again. The recovery ledger records the operator and recovery machine attestations, approval policy, source checkpoint, affected service families, new epoch, selected recipients, generated child bootstraps, acknowledgements, and first valid request proofs. Every child receives a unique new process and channel generation. The system can therefore restart a school, a cluster, or a data center from one human-authorized event without circulating a reusable fleet password.

A disaster policy defines exactly when a child may run and which duties remain held until additional review. For example, a fleet of web-cache agents may restart automatically; payroll settlement and credential exports may require a second human; signing authority may require quorum. A compromised service can be omitted from the replacement family even if it previously belonged to the fleet. Parent authorization never overrides the application owner's transaction limits or the service's own policy enforcement point.

Failure scenario 1: local utility power loss. The main data center loses electricity. Primary twins time out. The independent recovery machine remains available on its battery/solar supply, and its trusted ledger reference identifies the latest accepted state. When replacement workers boot, they enter a fresh attestation and per-machine admission sequence; stale proofs fail the epoch check.

Failure scenario 2: primary fiber outage with alternate communications. The main connection fails, but the recovery TEE has a cellular, satellite or separately provisioned private connection. The recovery authority verifies the alternative route against its own explicit policy, creates a disaster channel epoch, and issues targeted child bootstraps to reachable sites. Human browser clients with a strict route policy still require new human admission after route change.

Failure scenario 3: independent TEE process crash. A watchdog detects that the backup authority process terminated. Rebooting its operating system alone gives it no continuing channel proof. The protected machine identity and current recovery policy permit a fresh attested activation only under its designated tier-B rule. Failure of the machine attestation, tamper state or recovery material routes activation to the human-superuser tier.

Failure scenario 4: total loss of backup recovery devices. A designated administrator opens a separately registered human recovery entry point, proves control through the Human Admission Client, participates in any required co-approval, and establishes a new parent fleet epoch through the surviving root trust. Each child subsequently creates its own keys and proof stream, allowing hundreds of services to return under one approved organizational recovery decision.

Failure scenario 5: stolen child WASM circuit or cloned virtual machine. A copied binary or VM image reaches a different server. Its attempt to use the old circuit identity is bound to the previous machine, process generation and epoch. The service requires fresh machine evidence and route membership. Copying historical witness outputs provides consumed sequence positions. The cloned process must be separately admitted before it can participate in an authorized twin pair.

Failure scenario 6: stale database restored from backup. A previous ledger snapshot appears to forget spent proof positions. The service compares its state with independently retained witness/checkpoint or disaster-epoch evidence and holds operations when current state cannot be established. It resumes after state reconciliation or fresh authorized epoch transition. This is essential because runtime-bound proofs still require retained replay history to make consumption meaningful.

Failure scenario 7: malicious operator requests a huge fleet. The root policy limits delegated child count, allowed service images, destinations, resource budgets and permitted operations. A request exceeding its policy enters held review. The event is logged under the human identity and epoch, while each granted child remains independently auditable and revocable.

II.13.4 Recovery UI and operator workflow

The admission recovery console should present a compact disaster summary: affected fleet, unavailable service count, current recovery authority, machine inventory, last verified checkpoint, pending route changes, required approvers, and projected child instances. The operator selects a recovery policy, sees each privilege class, and approves the creation of a new fleet generation through the admission challenge. The console displays the new parent ID, issued children, successful attestation, channels established, held exceptions, and any incomplete business effects requiring reconciliation.

The experience distinguishes **authorization**, **deployment**, **channel establishment**, and **business readiness**. A human may approve the fleet and still have downstream workers unavailable because they lack power, accepted software, route access or a valid checkpoint. Each level is recorded separately. An offline approval package is sealed to its intended recipients and opened only by the authenticated machine, with the current policy epoch enforced when connectivity returns. A delayed package from an older epoch is refused.

II.13.5 Suggested policy profile

```
schema: http6.fleet-recovery.policy.v0
fleet: research-datacenter-west
root_admission:
  mode: human_admission
  roles: [fleet_superuser]
  approve_child_policy: true
  max_children: 512
  high_impact_actions: second_human_approval
ordinary_machine_restart:
  mode: attested_auto
  require_measured_runtime: true
```

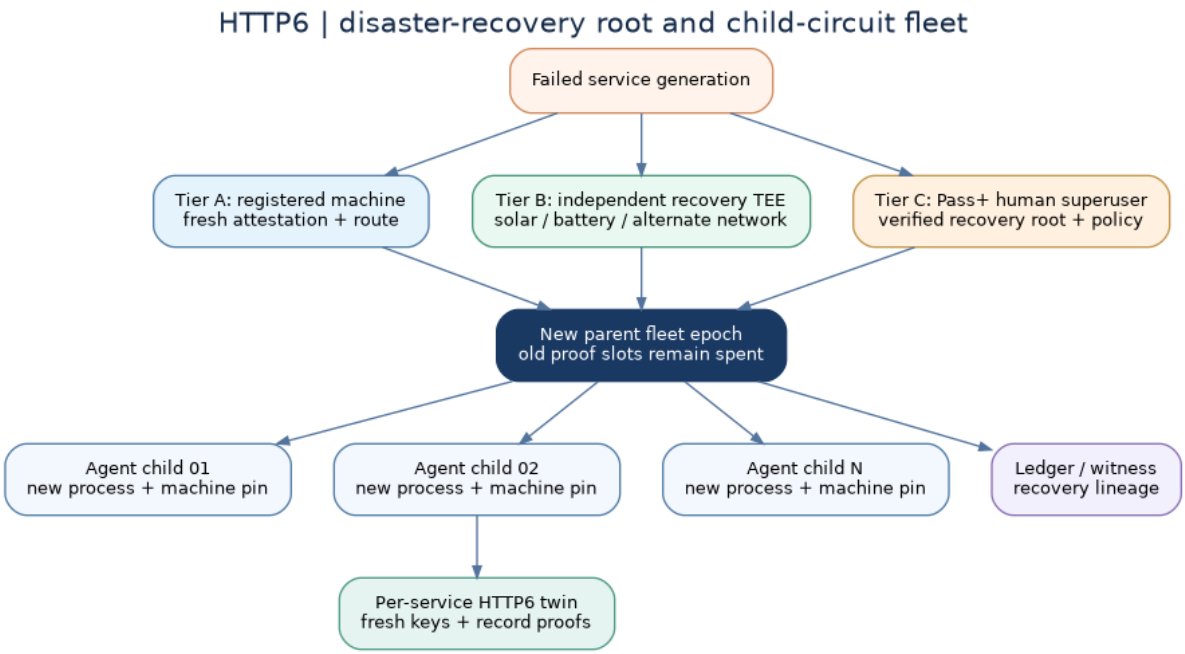

Figure 5: Three-tier recovery hierarchy: attested reboot, independent TEE and human recovery superuser, followed by fresh child generations.

```
  require_current_fleet_epoch: true
  approved_routes: [private_ingress, registered_service_endpoint]
independent_recovery:
  mode: pre_enrolled_tee
  power: autonomous_battery_solar
  alternate_network: authorized_disaster_link
  checkpoint: independently_retained
  issue_new_epoch: true
catastrophic_recovery:
  mode: human_superuser_ceremony
  recovery_root: pre_enrolled_offline_or_surviving_root
  create_new_parent_generation: true
  child_admission: machine_attestation_and_route_policy
  preserve_consumed_history: true
child_policy:
  per_machine_identity: required
  per_process_generation: required
  per_message_proof: required
  independent_revocation: true
```

This YAML is a source-level **illustrative policy schema**, not an already deployed service configuration. The deployment implementation must define signing, validation, authorization precedence, version migration, and storage semantics for each field. Any operator-side edit to fleet policy requires an authenticated administrative operation and an audit event.

II.14. Implementation work packages and direct deployment path

The program begins by specifying one authenticated request and one authenticated response through two synthetic, separately hosted executors. It then extends that primitive to a page with dependencies, a running browser session, cross-origin navigation, and a recoverable agent fleet. All packages share exact versioned protocol fixtures. The rollout deliberately establishes a small verifiable unit before expanding its execution and browser privileges.

WP-01 — Freeze the cryptographic and transcript specification. Select the supported KEM/key agreement, hash, HKDF, AEAD and MAC construction. Define canonical CBOR encoding, typed records, role and direction labels, key lifetime, rekey, protocol versions, authenticated service discovery, certificate or enrolled-attestation chain, and handshake transcript confirmation. Produce deterministic positive test vectors, corrupted-record vectors, and downgrade/reflection refusal vectors. The test must demonstrate that the client and server derive compatible directional secrets, and that different sequences, directions, and channel generations result in distinct record authentication contexts.

WP-02 — Human Admission Client to executor admission. Implement server availability and proof, trusted origin presentation, measured client process proposal, geometry challenge, consumed human decision, and twin process creation. Bind admitted runtime generation and service identity to the exact handshake. Verify a cloned executable, copied geometry response, forged server identity, changed route, and post-crash process all require fresh admission. Preserve saved personal circuits and existing Human Admission Client data during integration.

WP-03 — Wasmer WASIX HTTP6 service. Assemble a generic WASIX HTTP gateway with an admitted server circuit twin, resource budget, capability manifest, authenticated ProofCheck adapter, and durable ledger client. Deploy it to a controlled Wasmer Edge app using the current HTTP-server profile. Record the actual returned app URL, package identity, configuration and digest. Use a separate database or shared durable state service for nonce consumption and idempotency. Test instance stop/start, parallel app instances, invalid binary import, and wrong-tenant access.

WP-04 — HTTP6 browser workspace. Package the admission opening ceremony, channel adapter, HTML/content renderer, URL mapping, resource scheduler, manifest integrity checks, link interception, form submission, download handling and origin UI. Test a page containing styles, fonts, nested images, script-free interactions and link navigation. Introduce scripts only through a declared isolate that can neither bypass the HTTP6 scheduler nor directly invoke protected host operations. Run browser extension and native helper builds as distinct accepted products.

WP-05 — Agent executor and policy. Enroll one device, one measured WASIX executable and one exact service route. Start an authorized machine twin, verify every request and response, then reboot it. Exercise `attested_auto`, `human_on_reboot` and `break_glass_only` restart policies. Test wrong source address, attested-but-wrong executable, stolen process image, alternate IPv6 route and cloned binary.

WP-06 — Fleet admission and disaster recovery. Register a parent user, a modest test fleet, and one independently powered backup TEE. Qualify restoration from power loss, network isolation, alternate-network reentry, a failed primary checkpoint, and missing backup authority. Exercise human-superuser reconstitution against a known offline trust root, generate a fresh parent epoch and per-child admissions, and verify that a retired child cannot rejoin under its old proof chain.

WP-07 — Native transport. Implement native HTTP6 records on a controlled network with explicit local OS/browser integration. Match the `http6.n1` end-to-end confidentiality and integrity objectives independent of the outer HTTPS wrapper. Test active network interception, server impersonation, packet corruption, message replay, key compromise, forward-secrecy behavior and sustained load. Move one controlled asset/API service to a native HTTP6 endpoint while maintaining a separate conventional HTTPS service for incompatible clients.

WP-08 — Media and mesh. Begin with two endpoints and a relay; add three synchronized participants and a group membership epoch. Verify track provenance, per-packet or per-chunk proofs, out-of-order detection, slow consumer behavior, stream close, rekey, removal of one member and a reconnect requiring a new admitted generation. Publish latency and bandwidth measurements alongside successful secure behavior.

II.14.1 Milestones and acceptance conditions

Milestone	Implementation output	Required acceptance evidence
M0: source and protocol freeze	Exact codec, cryptographic suites, record profile and threat model	Byte-identical cross-language vectors; peer identity and downgrade refusal
M1: one-shot admission	Human admission proof instantiates one measured twin pair	Fresh geometry succeeds; replay and wrong-generation attempts fail
M2: encrypted oscillation	Request, response and heartbeat proofs on one live channel	Distinct proofs and keys; duplicated sequence consumed; crash retires generation
M3: WASIX web delivery	Gateway serves HTML, CSS, image and a second page	Each asset checked before rendering; resource and tenant isolation pass
M4: extension/native bridge	Customer browses in an HTTP6 workspace	Link and form navigation remain within the controlled scheduler
M5: agent/fleet	Attested machines, child delegation and reboot policies	Copied circuit and wrong route refused; revocation reaches descendants
M6: disaster recovery	Independent recovery TEE and human-superuser fallback	New epoch authorized, spent proofs retained, one fleet restart audited
M7: native channel	TLS-independent HTTP6 endpoint in a controlled environment	Active on-path and malicious endpoint tests; confidentiality/integrity acceptance
M8: scalable conference	Multi-recipient proof-bearing media	Authenticated track/packet identities, membership rekey, measured jitter

These are engineering completion gates and a build sequence. A source test, a deployed website, a proven cryptographic implication, and an accepted end-to-end channel are distinct forms of evidence. The release registry records each separately so product messaging identifies the exact accepted profile.

II.15. Operational security, monitoring, and recovery controls

A proof-based channel is operationally secure when the service can explain **who admitted it, which machine and process operated it, what resource it touched, what sequence was con-**

sumed, and which business effect completed. The HTTP6 observability schema therefore records opaque channel IDs, verified service identities, attestation decisions, parent/child lineage, policy epochs, sequence-advance summaries, route state transitions, and reconciliation references. It minimizes personal proof material, raw enrollment secrets, decrypted private circuits, traffic keys, and protected business payloads in general-purpose logs.

Operators monitor rejected handshake attempts, abnormal geometry failures, expired sequence positions, stalled heartbeats, route changes, failed attestation, rejected executable measurements, concurrent conflicting child starts, missing checkpoint members, rekey failures, and disaster epoch creation. An elevated rate of revoked proofs is a measurable symptom of replay attempts or client bugs; the remediation distinguishes those possibilities without exposing credential secrets in telemetry.

A controlled termination explicitly invalidates the generation and retires outstanding positions. A crash triggers remote suspicion and closes access after the heartbeat deadline. A recovered worker performs fresh admission, and business operations reconcile their original IDs. A human-controlled runtime reacquires an admission proof when strict policy requires it; a machine-controlled runtime uses an enrolled restart policy; a catastrophic fleet invokes one of the recovery tiers. Restored databases never treat a historical unused value as current without checking independently anchored consumption history.

Machine and route allowlists require ongoing updates as infrastructure changes. Vendor administrators enroll new IPv6 networks, planned proxy gateways, new WASIX release hashes, and changed hardware identity under governed policy. Changes produce a new policy epoch, which is recorded in the accepted channel and every new proof. An emergency proxy replacement cannot silently inherit a prior server identity simply because its hostname matches.

II.16. Open engineering decisions to fix in the protocol profile

A specification for first implementation must select a finite, reviewable parameter set: cryptographic suites and algorithm agility; exact peer-identity roots; hardware and TEE attestation format; proof-to-key-schedule binding; record codec; maximum message and chunk sizes; rekey thresholds; channel liveness timeouts; congestion, retransmission and ordering policy; persistent replay-state schema; acceptable clock skew; operating-system privilege model; and private-data retention. The same profile also fixes the exact route canonicalizer and policies for IPv4/IPv6 mobility. Selection is a product and engineering decision recorded in the release manifest rather than an untrusted on-the-wire default.

At the browser boundary, the initial renderer must decide which remote content types can execute code and how it enforces a content security policy. For the first milestone, an HTML/CSS/image/document subset with trusted link handling offers a measurable demonstration of secure web navigation. General-purpose JavaScript can follow after the renderer's network and permission interception is demonstrably complete. For a fully native browser, standards compatibility, accessibility, service workers, cache semantics, CORS/origin rules, and download protections deserve independent feature matrices.

At the recovery boundary, the design must specify custody of the offline root, last-accepted state preservation, approval quorum, authorized site list, how disaster epochs are propagated through isolated networks, the maximum child issuance permitted in an outage, and the exact test proving that retired authority cannot revive from a stale device image. An independently

powered laptop increases survivability only if its trust material, code, retained state and communications plan remain usable during the emergency.

II.17. Architectural conclusion

HTTP6 is proposed as a complete alternative web-channel architecture built around **human-admitted live circuits, independently authenticated service twins, per-message proof, synchronized encryption, WASIX-bounded execution, and policy-governed lifetime**. Its initial admission geometry ceremony gives a person the authority to open a particular measured runtime. Its oscillation gives that runtime a distinct cryptographic proof for each request, response, asset, media block and heartbeat. Its fleet model gives an administrator control over whole schools of agent runtimes while preserving each child's machine, route and operation identity.

A vendor can begin by exposing an HTTP6 gateway behind an existing web endpoint and gradually transfer resources to proof-enforced APIs. A controlled browser workspace makes the interaction visible. Native HTTP6 connections can later replace TLS on admitted channels by providing authenticated peer negotiation and encrypted record transport directly. The disaster-recovery hierarchy preserves operational continuity through machine-attested reboot, independent recovery TEE, and human-superuser reconstitution of a new fleet epoch. The resulting unit of trust is **a living, authenticated process making a specific request in the next verified channel state**, rather than a bearer artifact that remains useful merely because its expiration time lies in the future.

II. Appendix A. Vocabulary and source basis

HTTP6: proposed proof-native, proof-carrying web/API channel. **Human Admission Client:** user-facing admission and proof application. **HTTP6 reference architecture:** proof, custody, runtime, policy, authority and infrastructure services (Proof Authority, Enrollment Service, Minting and Delegation Service, Policy Decision Point, Authoritative State Plane, Consumption Ledger). **Rosario–Wang:** underlying private-reference proof construction. **Twin:** a pair of authenticated circuit executors maintaining synchronized channel state. **Oscillation:** continuing direction-specific exchange of authenticated, encrypted, fresh message records. **Human admission:** interactive local challenge required to create a specified process generation. **Child circuit:** a distinct policy-constrained machine or process issued under an authorized fleet parent. **Recovery TEE:** pre-enrolled protected recovery processor capable of creating a new recovery epoch under its assigned policy. **Disaster root:** surviving trust material and ceremony needed to establish fleet authority when regular recovery hardware is unavailable.

II. Public technical references

1. IETF, **RFC 9846**, *The Transport Layer Security (TLS) Protocol Version 1.3*, July 2026. https://www.rfc-editor.org/info/rfc9846/. Secure-channel requirements and engineering comparison.
2. IETF, **RFC 9334**, *Remote ATtestation procedureS (RATS) Architecture*, January 2023. https://www.rfc-editor.org/rfc/rfc9334.html. Attester, verifier, relying-party and freshness roles.
3. IRTF, **RFC 9180**, *Hybrid Public Key Encryption*, February 2022. https://www.rfc-editor.org/rfc/rfc9180.html. Example of a well-defined KEM/KDF/AEAD construction.

4. IETF, **RFC 9449**, *OAuth 2.0 Demonstrating Proof of Possession (DPoP)*, May 2023. https://www.rfc-editor.org/rfc/rfc9449.html. Comparison with existing sender-constrained request proofs.
5. Google Chrome for Developers, *Native messaging*. https://developer.chrome.com/docs/extensions/develop/concepts/native-messaging. Extension-to-native host boundary.
6. Google Chrome for Developers, *Declarative Net Request API*. https://developer.chrome.com/docs/extensions/reference/api/declarativeNetRequest/. Browser network interception and redirection constraints.
7. Google Chrome for Developers, *Deal with remote hosted code violations*. https://developer.chrome.com/docs/extensions/develop/migrate/remote-hosted-code. Packaged executable resources in Chrome extensions.
8. Wasmer, *Wasmer Edge Apps*. https://docs.wasmer.io/edge/learn/apps/. HTTP hosting and ephemeral application lifecycle.
9. NIST, *SP 800-34 Rev. 1: Contingency Planning Guide for Federal Information Systems*. https://csrc.nist.gov/pubs/sp/800/34/r1/upd1/final. Independent recovery, contingency and alternate facilities.

II. Document provenance

This is a **new architecture and engineering specification** prepared from the owner's description of the reference architecture and its Human Admission Client and supplied prior project materials. Public references document the surrounding transport, browser, attestation and recovery technologies. Internal source paths and implementation names in the source-layout sections are proposed module names, rather than claims that the completed HTTP6 protocol has been released. External vendors can adopt the specified profile in stages as each implementation milestone is accepted.

Part III. Executable Rosario–Wang attachment

The following mathematical and implementation appendix is reproduced from the supplied November 9 attachment. Its section labels and equation tags remain those of that source. Citation keys refer to the source register at the end of this part. Its original evidence statements retain their dates; the new reproduction is recorded in Part I and the regeneration evidence directory.

Abstract

This attachment converts the supplied finite Rosario–Wang formalizations into a runnable HTTP6-style demonstration. The code implements private enrollment, calibrated geometric membership, fresh additive masks, exact designated verification, authorized recovery, and an observer-visible record. New, explicitly identified engineering components bind that record to an executing instance, peer, application operation, inherited policy, and one-use event. A synthetic human admission activates an independently credentialed root; an authorized mint creates a distinct child; requests and responses carry fresh native membership proofs. Exact finite enumeration checks the released distributions and private decoding, while a symbolic resource calculation explains what the sequential privacy theorem means at one trillion observations. The executable reports its actual bounded tests separately from mathematical implications and production implementation obligations.

A. The implemented statement and private witness

The source uses commitment in the sense of an enrolled value and private dictionary. Define $X = (k, \varphi)$, where k is the enrolled string and φ is the private permutation of the response labels. The demonstrator retains this object in the designated private enrollment model. Its public identifier is an opaque instance or account handle, not an unsalted digest of a guessable secret. A separate cryptographic commitment scheme, with opening randomness and hiding/binding proofs, is not introduced by renaming that handle. [S1, §§1, 11–12; S2, §19.4]

For event e, the public statement x_e names the profile, complete event identity, nonce, frames, schedule, sender, intended verifier, channel, direction, stream, sequence, instance configuration, policy and lineage digests, operation identifier, exact synthetic payload, and expiration. The proof object contains this statement, the masked response word, and a separately keyed envelope authenticator. The actual private mask allocation is a protected setup resource, not a freely chosen existential variable. [S1, B1, B4; S2, E08]

$$w_t = \varphi(c_t(k_{r(t)})), \qquad z_t = (w_t + m_t) \bmod H. \tag{A1}$$

The cell function c_t identifies which public partition cell contains the selected character. The dictionary maps that cell to a private response. A fresh independent mask $m_t \in \mathbb{Z}_H$ is added before release. `raw_word`, `release`, and `respond` implement these operations. Calibration coordinates precede the data coordinates and use their own freshly allocated masks. This is the source primitive executed in each record, not a placeholder calling an unrelated proof library. [S1, Equations 1, 7, K2]

The designated verifier's native predicate is

$$\mathsf{NativeValid}_e \iff \big(\forall t:\ \varphi^{-1}((z_t - m_t) \bmod H) = c_t(k_{r(t)})\big) \wedge \mathsf{CalibrationValid}_e. \tag{A2}$$

`verify_native` checks the equivalent complete unmasked response word against the enrolled reference. A separate `recover` function takes only public profile information, the masked word, and actual masks. It reconstructs the dictionary from calibration and then the string from separating cell signatures. It never reads the reference store. That makes authorized recovery inspectable as a separate operation from comparison against enrollment. [S1, §§3–5; S2, T-03, T-05]

B. Geometry selection and the source profiles

The four-symbol ring implements the source's two offsets $(\nu, \nu+3) \bmod 4$ and its full two-response calibration. RW-12/4 implements the source's twelve distinct unequal coordinate pairs in $\mathbb{Z}_4^2$, nonce-indexed cyclic assignment, and three-response calibration. The 40-symbol profile implements the source's exact triple set

$$\mathcal{Q} = \{(a,b,c) : (c-a) \bmod 4 \in \{0,1\}\} \cup \{(a,b,c) : (c-a) \bmod 4 = 2,\ (b-a) \bmod 4 \in \{0,1\}\}. \tag{A3}$$

This set contains 40 distinct triples. Each coordinate gives a balanced four-cell facet, and the three coordinates separate all assigned symbols. The code fixes digits followed by uppercase letters as the 36 live symbols and uses four explicitly designated pads. A new nonce-dependent cyclic reindexing is an engineering extension: it permutes assignments of symbols to the source triples, so neither balance nor separation changes. The exact triple family and its separating argument are source-derived; the additional nonce adapter is identified as new. [S2, E44; S5, Lemma 16]

The public `challenge` command prints all cells for inspection. Private cell selections, dictionaries, and masking values are kept out of the observer export. This is a software presentation of the specified challenge interface; a protected display/input device is an implementation boundary to be supplied rather than simulated as a physical security result. [S1, §3; S3, A7]

C. Private entropy as a counted circuit resource

The finite tape is instantiated as a byte buffer. It is allocated once and consumed in disjoint bit positions. Each event receives a complete mask bundle before its challenge is released. Allocation advances a monotonic cursor, even if the event is later rejected, cancelled, or expires. Exhaustion stops issuance. Equal numerical mask values at different positions remain valid independent realizations in the ideal model; reusing the same random allocation is the excluded dependence. [S1, §§7–8]

For fixed cell count H and calibrated response count q, the exact independent-mask profile consumes

$$E_{\text{record}} = q \log_2 H, \qquad E_N = Nq \log_2 H. \tag{A4}$$

RW-12/4 uses $q = 2\ell + 3$. The implemented RW-40 profile uses $q = 3\ell + 3$. A 12-character keypad event therefore uses 54 ideal private bits. A 24-character machine event in RW-40 uses 150. At one trillion events those become 6.75 TB and 18.75 TB of continuously packed private mask material respectively. These quantities exclude provisioning, authentication keys, record metadata, durable state, and application data. One trillion request/response pairs use twice the per-record quantity. The calculation follows directly from the fixed encodings and counts, not a measured trillion-event run. [S1, E4, §10.3; S2, §14.1]

The runtime obtains provisioning bytes through the operating system's cryptographic random source. Exact enumeration separately ranges over the ideal finite mask distribution. These are two explicit evidence modes: arithmetic verification of the model and practical sampling for the executable demonstration. A production exact-privacy claim establishes the relevant private distribution and provisioning assumptions. Generating unlimited masks from a short seed instead establishes a computational profile, with its own analysis. [S1, E5; S2, §14.1]

D. The observer's posterior, including public code

Let B include the public program, permitted profile definitions, public instance records, fixed credential length, and admitted side information. For the native transcript, let G_e be the complete credential-independent public challenge and Z_e the combined calibration and data responses. Under the source's fresh private-mask model,

$$\Pr[Z_e = z \mid X = x, B, G_e = g] = H^{-q}. \tag{A5}$$

For each candidate credential, exactly one mask vector produces a given released word. Every vector has the same probability, so every candidate has the same observation likelihood. Public knowledge of the code does not change that count. The observer can compute many compatible explanations; those computations do not identify the historical mask or credential. The actual masks select the authorized private slice of the observation fiber. [S1, Equations 3–6; S3, Theorems 2 and 4]

The finite audit enumerates the complete RW-12/4 output law for all 288 one-symbol/dictionary pairs at a fixed nonce. Each pair produces all 1,024 possible five-coordinate outputs exactly once as its mask ranges over the 1,024 possibilities. It also checks the binary ring over all sixteen candidate nonces, two complete consecutive binary events, and calibrated RW-40 recovery. An exact `Fraction` calculation preserves a nonuniform prior, avoiding the mistaken inference that privacy forces every credential to become equally likely.

Sequential freshness gives

$$\Pr[\mathcal{T}_{1:N} = t_{1:N} \mid X = x, B] = \prod_{e=1}^{N} Q_e(g_e \mid B, t_{<e}) H^{-q_e}, \qquad I(X; \mathcal{T}_{1:N} \mid B) = 0. \tag{A6}$$

The public challenge kernels Q_e must be credential-independent given the prior public history, and each event must have the required fresh private masks. The product then contains no dependence on x; Bayes' rule preserves every supported prior. This source theorem supplies the

trillion-observation result. The program records that result as a **conditional mathematical consequence**, never as a trillion-logon benchmark. [S1, S1–S4; S2, E50]

The full runtime export additionally contains application metadata, real wall-clock fields, and computational tags. The exact finite audit fixes its public controls and does not establish secret-independent timing for Python or privacy for attacker-driven errors. The envelope's independent key and deterministic processing of the permitted transcript provide a compositional design path; a complete deployment analyzes all additional emissions jointly. If the acquired object includes the actual masks or enrolled secret, its observation model is different: the supplied recovery function demonstrates the resulting disclosure. [S1, S5–S6; S3, P5–P6]

E. Enrollment, origin, and application authentication

The native masked word does not by itself bind an arbitrary HTTP body. This attachment adds a **new engineering envelope**, separate from the source privacy theorem:

$$\tau_e = \mathsf{HMAC}_{a_v}(\mathsf{Enc}(\texttt{witness-envelope}, x_e, Z_e)) \,. \tag{A7}$$

The key a_v is provisioned independently of the human commitment and mask tape. The client checks a domain-separated challenge authenticator before applying its private material. The verifier compares the complete statement with the authoritative issued event and checks the independently supplied binding. That binding identifies sender, recipient, channel, direction, and stream. It is not accepted merely because an untrusted request writes those values in its own header.

Changing the target or payload changes the authenticated statement; using the record on a different bound channel fails comparison. The local component also checks the intended payload and admitted instance before responding. Tests cover these substitutions. Actual hardware-backed attestation, protected confirmation, and resistance to a physically relayed screen remain the deployment's trusted-context and interaction requirements. This executable checks their logical interface; it does not report a real hardware quote or a trusted-screen experiment. [S1, §§12.4–13; S6, action context and live-process authentication]

Envelope authenticity uses computational HMAC security. The source's zero credential information is a different proposition. The demonstration also leaves synthetic application payloads readable; protected production transport or reviewed authenticated encryption supplies confidentiality of arbitrary web contents. These distinctions allow an attachment to demonstrate a real native witness without overloading that witness with unsupported transport guarantees.

F. One-use validation and the local business effect

The verifier checks packet structure, exact stored context, live epoch, deadline, and envelope authenticity before claiming an event. Its first eligible complete response atomically transitions `ISSUED` to `BURNED`. It then performs native credential verification. A correctly authenticated envelope carrying wrong native responses consumes the event and is rejected. Repetition encounters `BURNED`; random invalid tags do not obtain the native-verification claim. [S1, B1–B4]

$$\mathsf{Accept}_e = \mathsf{ContextValid}_e \wedge \mathsf{Claim}_e \wedge \mathsf{CalibrationValid}_e \wedge \mathsf{DataValid}_e \wedge \mathsf{PolicyAllows}_e. \tag{A8}$$

The local SQLite implementation commits the burn before subsequent verification. Its effect table enforces a unique `(sender,operation_id)` pair and compares the canonical payload on reconciliation. A fresh valid request for the same business operation returns the previously recorded outcome; a changed payload under the same operation identity is rejected. The effect being established is a local table insertion. External transactions require an explicit idempotence and reconciliation contract with their own effect boundary. [S1, §13.3]

The test suite races multiple threads and two separate database connections for the same claim, reopens a consumed database, and retires unfinished events during restart. These are actual local observations. They are not a distributed consistency proof or protection against arbitrary rollback of the complete host.

G. Human origin and per-message machine proofs

The synthetic human's accepted ceremony activates a root machine instance. That root already has independent machine reference material. Its authorized `MINT` message creates a distinct child under a fixed approved template. The child receives neither the human credential nor a deterministic derivation of it. Every request subsequently uses the child's own calibrated membership relation and private tape. Every response uses the service's independent relation and private tape.

Effective permission remains an intersection along the registered ancestry:

$$\mathcal{A}_v = \bigcap_{u \in \mathrm{Anc}(v) \cup \{v\}} \mathcal{A}(P_u). \tag{A9}$$

The fixed reader template permits the approved report read and forbids export and further minting. The registry authenticates policy content and parent-grant linkage. A changed policy object fails its original grant check. Parent revocation is checked again after native verification and before effect creation. Root activation and minting are reached through admitted messages rather than through receipt of a public grant alone. The registry uses an independent issuer MAC in this attachment; a publicly verifiable deployment uses its specified digital-signature and trust-distribution profile. [S6, §§6.4–6.6]

This demonstrates the desired continuity: a human decision authorizes automation, while the automated messages carry fresh native proofs under distinct references. It also illustrates why privacy alone does not establish genealogy: the registry and policy gate supply the provenance and permission decisions in addition to the masking construction.

H. Efficiency and quantified experiments

For fixed alphabets, `respond` and `verify_native` evaluate a constant number of table operations per enrolled character. Calibration contributes constant overhead. `recover` uses a fixed separating-code inverse after privately recovering the dictionary. The total primitive work is $O(\ell)$ for these profiles. It does not search a candidate space or process all historical records. Full message handling additionally performs canonical encoding, HMAC, ancestry checks, and indexed event/effect operations.

The supplied benchmark measures 10,000 core records and reports actual elapsed time, Python version, and platform. It explicitly excludes the transport, envelope, ledger, hardware, and entropy-provisioning work. The test logs and JSON results supplied alongside this appendix are the authoritative measured values; no result is described as a throughput guarantee for the eventual protocol.

The source's distinct restricted blind-injection experiment gives

$$\Pr[\text{any acceptance in } Q \text{ eligible fresh attempts}] \leq \min\{1, QH^{-q}\}. \tag{A10}$$

That experiment gives the attacker no current masks, no current valid response, and no release oracle for the target event. The 24-symbol RW-40 profile has $q = 75$, so $H^{-q} = 2^{-150}$. With $Q = 10^{12}$ restricted attempts, the union bound is about 7.01×10^{-34}. This is not an unrestricted impersonation theorem or a bound on every MITM. It is printed with the precise access conditions that support it. [S1, A1]

I. Reproduction and evidence status

Run the demonstration, finite privacy audit, capacity calculation, and unittest suite using the commands in `README.md`. The package includes public sample records, a record schema, source hashes, a function-to-equation map, measured reports, and this Markdown attachment. The script is complete for the stated local experiment and has no undefined primitive in its forward response, designated verification, or private extraction path.

The executable additions are new reference code authored for this review task. The mathematical source establishes the conditional claims; the Python tests establish the reported finite behavior. The supplied Lean registry provides exact declaration names but explicitly distinguishes source correspondence from a new kernel run. The separate exact-projection paper is not used to claim polynomial SAT solving or computational hardness of passive credential recovery. This attachment needs neither conclusion. [S4; S5]

Source register

[S1] `Rosario_Wang_ENI6MA_Consolidated_Formalization.md`, 16 September 2026. Primitive, ring/keypad construction, calibrated recovery, nonce/mask entropy, event burn, sequential privacy, and restricted active game.

[S2] `ROSARIO-WANG-COMPLETE-FORMALIZATION(2).md`, 14 September 2026. Finite incidence encoding, private reference, calibration extension, exact 40-symbol facets, and information/resource accounts. The separate unnumbered attachment could not be read or recovered in this run; no byte identity is assumed.

[S3] `ENI6MA-Unified-Geometric-Formalization.md`, 13 September 2026. Candidate intersections, public fibers/private slices, conditional privacy, and additional-observation analysis.

[S4] `B-lean-formalization-registry(1).md`. Theorem/source correspondence, including modular release, privacy, separation, joint extraction, and the stated evidence boundary.

[S5] `ROSARIO-WANG-EXACT-PROJECTION-FULL-FORMALIZATION.md`, version 1.1, 14 September 2026. Exact three-facet chamber construction in §17; separate polynomial-compression obligation is not used by this attachment.

[S6] `ENI6MA_White_Paper_Human_Originated_Identity.md`, 8 November 2026. Prior supplied architecture for action-bound MACs, synchronization, inherited policies, and resource checks. The implementation choices here remain explicitly identified engineering glue.

[E1] Python official `secrets` documentation: https://docs.python.org/3/library/secrets.html

[E2] Python official `hmac` documentation: https://docs.python.org/3/library/hmac.html

[E3] SQLite official transaction documentation: https://www.sqlite.org/lang_transaction.html

Appendix A. Complete executable-envelope schema

This is the unchanged JSON Schema for the delivered local review envelope. Its representation constraints are supplemented by the canonical codec, exact stored context, designated masks, current authority, and ledger checks. The schema does not itself supply an authenticator or define the final N1 wire encoding.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:http6:review:rw-envelope:1",
  "title": "HTTP6 Rosario-Wang review envelope (not an adopted HTTP6 wire format)",
  "description": "Representation validation only. Stored context, H-dependent residue ranges,
  ↪  canonical bytes, policy, masks, state, and authenticity are checked separately.",
  "type": "object",
  "additionalProperties": false,
  "required": [
    "statement",
    "released",
    "proof_tag"
  ],
  "properties": {
    "statement": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "epoch",
        "event",
        "sender",
        "recipient",
        "channel",
        "direction",
        "stream",
        "device",
        "measurement",
        "operation_id",
        "protocol",
        "kind",
        "sequence",
        "nonce",
        "issued_ns",
        "expires_ns",
        "policy_digest",
        "lineage_digest",
        "request_ref",
        "payload",
        "profile",
        "length",
        "calibration_cells",
        "responses",
        "frames"
      ],
      "properties": {
        "epoch": {
          "type": "string",
```

```
          "minLength": 1
        },
        "event": {
          "type": "string",
          "minLength": 1
        },
        "sender": {
          "type": "string",
          "minLength": 1
        },
        "recipient": {
          "type": "string",
          "minLength": 1
        },
        "channel": {
          "type": "string",
          "minLength": 1
        },
        "direction": {
          "type": "string",
          "minLength": 1
        },
        "stream": {
          "type": "string",
          "minLength": 1
        },
        "device": {
          "type": "string",
          "minLength": 1
        },
        "measurement": {
          "type": "string",
          "minLength": 1
        },
        "operation_id": {
          "type": "string",
          "minLength": 1
        },
        "protocol": {
          "const": "http6-rw-review-1.0"
        },
        "kind": {
          "enum": [
            "ADMISSION",
            "MINT",
            "REQUEST",
            "RESPONSE"
          ]
        },
        "sequence": {
          "type": "integer",
          "minimum": 1
        },
        "nonce": {
          "type": "integer",
```

```
          "minimum": 0,
          "maximum": 340282366920938463463374607431768211455
        },
        "issued_ns": {
          "type": "integer",
          "minimum": 0
        },
        "expires_ns": {
          "type": "integer",
          "minimum": 0
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "lineage_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "request_ref": {
          "type": [
            "string",
            "null"
          ]
        },
        "payload": {
          "type": "object",
          "required": [
            "operation",
            "resource",
            "destination"
          ],
          "properties": {
            "operation": {
              "type": "string",
              "minLength": 1
            },
            "resource": {
              "type": "string",
              "minLength": 1
            },
            "destination": {
              "type": "string",
              "minLength": 1
            }
          },
          "additionalProperties": true
        },
        "profile": {
          "enum": [
            "ring4",
            "rw12",
            "rw40"
          ]
        },
```

```
        "length": {
          "type": "integer",
          "minimum": 1,
          "maximum": 1024
        },
        "calibration_cells": {
          "type": "array",
          "items": {
            "type": "integer",
            "minimum": 0,
            "maximum": 3
          },
          "minItems": 2,
          "maxItems": 3
        },
        "responses": {
          "type": "integer",
          "minimum": 4,
          "maximum": 3075
        },
        "frames": {
          "type": "array",
          "minItems": 2,
          "maxItems": 3,
          "items": {
            "type": "array",
            "minItems": 2,
            "maxItems": 4,
            "items": {
              "type": "string",
              "minLength": 1
            }
          }
        }
      }
    },
    "released": {
      "type": "array",
      "minItems": 4,
      "maxItems": 3075,
      "items": {
        "type": "integer",
        "minimum": 0,
        "maximum": 3
      }
    },
    "proof_tag": {
      "type": "string",
      "pattern": "^[0-9a-f]{64}$"
    }
  }
}
```

Appendix B. Payload-case acceptance register

Each scenario generates fresh synthetic enrollments, contexts, masks, and proofs. Expected failures are successful tests when the exact required rejection and effect count occur. Complete input JSON and newly observed artifacts are in the validation and evidence directories.

Case	Scenario	Expected decision	Local effects
P01	Fresh request and separately proven response	`ACCEPTED`	1
P02	Body substitution invalidates the original statement	`REJECTED_EXCEPTION`	0
P03	Destination substitution invalidates the original statement	`REJECTED_EXCEPTION`	0
P04	Repeated consumed record creates no second effect	`REPLAY`	1
P05	Authenticated export still fails inherited policy	`REJECTED`	0
P06	Verifier rejects another channel	`REJECTED_EXCEPTION`	0
P07	Local participant rejects a relayed context	`REJECTED_EXCEPTION`	0
P08	Wrong proof direction is rejected	`REJECTED_EXCEPTION`	0
P09	Invalid envelope tag does not consume an eligible native attempt	`REJECTED_EXCEPTION`	0
P10	Wrong native word burns the authenticated attempt	`REJECTED`	0
P11	Retired parent fences a prepared valid proof	`REJECTED`	0
P12	Changed child policy fails original admission	`REJECTED`	0
P13	Leaf circuit cannot create an authorized grandchild	`REJECTED_EXCEPTION`	0
P14	Fresh proof for same business operation reconciles	`RECONCILED`	1
P15	Same business identifier cannot conceal changed content	`OPERATION_CONFLICT`	1
P16	Response naming an unrelated recipient fails request linking	`REJECTED`	1

Appendix C. Schema, model, and source-code inventory

Asset group	Complete supplied files
architecture	`Authority_Policy_Schema.json`; `protocol_contracts.schema.json`; `review_objects.schema.json`
assurance examples	`Authority_Policy_Example.json`; `Authority_Policy_Example.yaml`; `additional_classification_examples.json`; `canonical_test_vector.json`; `csf_outcome_examples.json`; `cui_family_examples.json`; `family_examples.json`; `review_action_proof.json`; `review_child_grant.json`; `review_evidence_record.json`; `review_policy.json`; `review_public_registry.json`; `review_request.json`; `review_root_grant.json`
models	`architecture.mmd`; `state_machine.yaml`
review runtime	`http6_review_envelope.schema.json`; `validation_suite.schema.json`
test payloads	`native_vectors.json`; `validation_cases.json`

The full contents of each listed file are supplied in the ZIP. The generic policy and protocol-contract schemas are architecture/reference artifacts distinct from the executable envelope schema. The expanded review volume also embeds its source code, schemas, and example catalog as appendices. Reproduction instructions remain inside each preserved standalone package.

Appendix D. Submission and authorship controls

The owner-supplied author and advisor credits are carried forward in the current submission front matter and revised IEEE manuscript. The original formalizations, architecture, and evidence papers retain their original identities. No affiliation, ORCID, assent, signature, funding statement, or correspondence address has been invented. Before sending, the authorized submitter confirms contact and affiliation information, coauthor/advisor approvals, the public or restricted disclosure scope, and any applicable contribution terms.

The cover letter requests technical review and a suitable demonstration path. It does not represent an accepted NCCoE collaboration, a CMVP application, a registered HTTP version, a publication acceptance, or an executed transmission. The native-profile decision register and the NIST requirement/evidence assignments accompany the request so that the receiving team can see both the available evidence and the work it is being asked to discuss.

HTTP6: From Durable Credentials to Continuous, Hereditary Authority

Frank Dylan Rosario, Lin Wang, Octavia Sulea, Francisco Perez,
Dawn Lipscomb, Maxwell Calixto, and Robert Bloodgood III

***Abstract*—HTTP6 proposes a secure application-channel architecture in which authority remains connected to its human origin throughout automated execution. The Human Admission Client provides the human-facing approval ceremony, the Rosario–Wang construction supplies the private-reference admission relationship, and the HTTP6 reference architecture supplies distinct runtime identities, protected key use, constrained delegation through its Minting and Delegation Service, and resource-level enforcement through its Policy Decision Point. The proposed protocol binds requests and responses to the admitted instance, intended peer, operation, policy, and current message state. This paper develops that model through four formal expressions and thirty inline worked examples. It explains how action-specific proofs and one-use state address transferable bearer authority, how authenticated context and protected local participation address phishing and cross-device screen relay, and how permission intersection preserves the original authorization ceiling across child processes. It also examines device-bound geometry, conditional observation privacy, anonymous and pseudonymous access, multifactor profiles, controlled recovery, and compatibility with existing web infrastructure. The mathematical arguments retain their stated premises, and the examples illustrate the behavior specified by the surrounding theory. The resulting contribution is a unified account of human, agent, and service communication in which copied artifacts remain records of prior activity rather than independent authority for future action.**

***Index Terms*—HTTP6; human admission; Rosario–Wang; hereditary authority; agent identity; phishing resistance; hardware binding; private-reference authentication.**

Worked-example edition — theory text with inline illustrations. The original theory, section order, equations, and reference register are retained. The numbered worked examples are added illustrations of the nearby argument, not recorded implementation results. Names, amounts, identifiers, and workflows in the examples are hypothetical; each example uses the conditions and enforcement mechanisms stated in its surrounding section. The source front matter identifies the original theory edition.

I. A web protocol organized around the right to act

Secure web communication rests on the Secure Sockets Layer (SSL) work developed at Netscape in the mid-1990s. In April 1995, Dr. Taher Elgamal presented "The Secure Sockets Layer Protocol (SSL)" at the Danvers IETF meeting, identifying endpoint authentication, traffic confidentiality, and data integrity as core objectives. [1] The IETF archive also identifies Elgamal and Kipp E. B. Hickman as authors of the April 1995 SSL Internet-Draft. [2] Paul Kocher's first-person account records Elgamal's assembly, in 1996, of the team comprising Kocher, Alan Freier, and Phil Karlton that produced SSL 3.0. [3] The November 18, 1996 specification, later preserved as RFC 6101, names Freier, Karlton, and Kocher as its authors and Elgamal among its contributors. [4]

That history informs the present proposal. Elgamal's 1995 presentation described per-record authentication and sequence-based freshness while identifying access-control and authorization schemes as additional Internet-security concerns. [1] HTTP6 develops the application-authority question: which human-approved process may invoke a specified operation, through which descendants, and under what continuing policy. Its proposed contribution joins that authority to each protected request and response, rather than treating secure-channel establishment as the complete authorization decision. The SSL sources are cited as historical foundations for that distinction.

The central question for an increasingly automated Internet is no longer simply whether two computers can exchange encrypted information. It is whether the software requesting an action presently possesses legitimate authority for that particular action. A person may authorize a task, an agent may interpret it, a service may divide it among workers, and those workers may invoke additional services. Throughout that progression, the original permission must remain recognizable, bounded, and revocable. HTTP6 proposes making this continuity part of the communication protocol itself, rather than leaving every application to reconstruct it independently.

In this proposal, HTTP6 is the working name for an architecture in which protected application messages carry evidence of authorized execution. The Human Admission Client supplies the human interface. The Rosario–Wang construction supplies the private-reference admission relationship. The Proof Authority, the Minting and Delegation Service, the Policy Decision Point, and the Authoritative State Plane supply the protected runtime, instance identity, policy enforcement, delegation, and synchronized state. HTTP6 connects those components through a common application-channel model. The proposal concerns an architectural direction, with formal consequences established under specified cryptographic and enforcement premises; the name denotes the project rather than an assigned Internet protocol version. [5]

Its organizing principle is straightforward: a successful login begins an accountable relationship instead of dispensing an object that can independently stand in for the person. The admitted process receives authority for defined work. Each child receives a distinct identity and a constrained delegation. Each protected request demonstrates its current relationship to that authority. A response demonstrates the corresponding service's participation and binds its result to the request. Human

Advisor and contributor: Dr. Taher Elgamal.

approval, machine execution, and communication consequently become connected stages of one verifiable operation.

Worked example 1 — One approval, several bounded steps.

Maya asks her research assistant to read three public papers and save a summary in her private workspace. The Admission Presenter displays that task for approval. The admitted root creates a reading worker and a summarizing worker, each with its own identity. Reading the papers and saving the summary fit the approved task. Sending the workspace to an unrelated recipient does not. Under the described enforcement rule, that attempted export is rejected even when it comes from a worker that legitimately participated in the research. The original approval follows the work without becoming permission to do anything Maya herself could do.

This could make HTTP6 especially suitable for traffic shared by people and agents. People need understandable consent, private credentials, convenient sessions, and reliable recovery. Agents need automatic operation, explicit tool permissions, predictable delegation, and machine-verifiable outcomes. Both need the same fundamental guarantee: information received from another party cannot silently acquire more authority than the receiver has deliberately granted. HTTP6's proposed advantage is to express that guarantee consistently across browsing, APIs, workloads, and delegated action.

Worked example 2 — A person and an agent use the same permission boundary.

Maya clicks a button to retrieve an approved report. Later, her scheduled agent retrieves the same report under a separate delegation. Both requests identify the particular report and the participant asking for it. The service checks each participant's current permission. The agent does not have to pretend to be Maya, and Maya does not have to approve every automatic retrieval. Their interaction differs, but the resource still asks the same question: is this participant authorized to perform this operation now?

II. The durable-secret problem is a problem of transferable authority

A durable credential becomes dangerous when possession of its copied representation is sufficient to exercise its permissions. The defining property of a bearer token is precisely that a holder can use it without demonstrating possession of an associated cryptographic key. Protection in storage and transit is therefore essential. The fundamental risk is not merely that the string exists for a long time; it is that the system accepts the string as portable authority. [6]

Imagine a build worker authorized to fetch a dependency. Giving it a reusable credential accepted across an entire repository creates a larger power than the immediate task requires. Copying that credential may reproduce the power elsewhere. Rotating it later can shorten exposure, but rotation leaves the same basic authorization rule in place: whoever presents the current credential may act within its scope. HTTP6 instead makes the requesting instance, present channel, intended operation, current policy, and unused authorization position jointly necessary.

Worked example 3 — A copied credential versus a copied request.

In a bearer-only example, a build worker sends a reusable repository token, and a thief who copies that token may use its remaining permissions from another machine. In the HTTP6 example, the worker sends a protected request for dependency D7, bound to worker W4, its current channel, and an unused action position. Copying the request does not supply the worker's protected key or a new action position. Once D7's request is consumed, replay produces no second authorized effect. Changing D7 to a private repository also changes the statement that requires a valid proof.

This distinction applies to passwords, API keys, session objects, refresh credentials, and some service-account configurations, but it does not treat every persistent key as defective. A protected device key may be a useful long-lived anchor. A recovery authority may need continuity across years. An encrypted archive may require retained decryption capability. The proposed transformation is to confine such secrets to controlled uses and prevent ordinary application artifacts from becoming unrestricted substitutes for their owners.

Accordingly, HTTP6 should be described as removing durable bearer authority from protected application traffic, not as abolishing cryptographic keys. It preserves keys where they establish identity, confidentiality, or recovery, while narrowing how applications invoke them. A child receives access to approved operations rather than a parent's signing secret. A recorded message becomes evidence of a previous event rather than permission for the next event. That is a structural change in authorization, not simply another secret-storage convention.

Worked example 4 — Keeping a useful key without handing it to every worker.

A document service keeps its long-lived decryption key inside a protected key service. A reporting worker is allowed to request decryption of the approved reporting view, but cannot export that key or ask for an unrelated archive. Copying the worker's configuration therefore does not reveal the decryption key. The key still exists and still performs useful work; access to its operations is restricted by the admitted identity and policy. This illustrates the difference between retaining a protected secret and distributing a reusable secret as every application's access pass.

The practical contrast appears in a stolen deployment archive. In the proposed model, the archive may reveal source code, configuration, and public delegation records. Those materials explain how the worker was authorized, but its new execution still needs an admitted instance, protected key access, and current state. The attacker must compromise the enforcing relationship rather than merely locate a familiar environment variable. Where a legacy backend still accepts an independent password, migration must remove that alternative path or keep it behind an independently protected boundary. The guarantee follows the actual resource gate, not the label attached to the frontend.

The format of a credential is secondary. A JWT can represent a bearer assertion or participate in a proof-of-possession design. DPoP already binds OAuth tokens to a client key, and current OAuth security guidance recommends sender-constraining access tokens where applicable. HTTP6 therefore builds its case on a wider integration of identity, lineage, policy, messages, and effects rather than on an assertion that existing systems universally depend on bearer possession. [7], [8]

III. From the human pilot light to independent machine identities

The human pilot light is the initiating decision that brings a root circuit into an authorized state. The person approves a defined service, task, privilege ceiling, and lifetime through the Human Admission Client. The private-reference verifier checks the required relationship and issues an authenticated admission for the exact root instance. That admission records the authorization's origin without copying the person's remembered commitment into every process that follows. The human remains the source of the approved authority, while software receives responsibility for exercising a limited part of it. [9]

Consider a researcher authorizing an agent to collect public papers and prepare a private summary. The root may read specified sources and write into one workspace. It may create document-processing children, but none receives permission to publish the researcher's files or change account settings. The root receipt identifies the approved undertaking; each birth record identifies the child, its key, executable configuration, policy, and parent. The architecture can explain the work's provenance without pretending that every child is the human.

Worked example 5 — Following a child's provenance without revealing the human secret.

Maya approves root R1. R1 admits collector C1, and an authorized mint admits summarizer C2 under C1's narrower grant. C2's record names its own public key, its parent record, and its permission to write the summary. A verifier can follow C2 to C1 and then to R1's human-admission receipt. None of these child records contains Maya's remembered commitment. The records explain why C2 may act, while C2's fresh key-possession proof establishes that the requester is the admitted child rather than somebody holding a copy of its public record.

This distinction matters for accountability. An agent's request should say, in effect, that this identified worker is acting under this approved delegation. It should not impersonate a continuously present person. A human authentication time and a workload's current activity are different facts. NIST's session guidance likewise distinguishes session continuity from renewed human authentication and warns against interpreting an access token as evidence of the subscriber's current presence. [10]

The human therefore authorizes meaningful scope rather than approving every network packet. Routine requests can be produced automatically by the admitted process. A sensitive operation, such as changing a payment destination or increasing an agent's budget, can require a new human ceremony. Convenience and control reinforce each other when the protocol preserves that boundary: automation operates freely inside its grant, while expansion returns to an accountable approval path.

Worked example 6 — Automatic work stops at the edge of the approval.

Maya approves a worker to prepare a report during a defined one-hour task. The worker can fetch approved inputs and save drafts automatically within that grant. Publishing the report publicly is a separately protected operation requiring a new human confirmation. If the worker requests publication, the system holds that operation for approval instead of treating the earlier research login as publication consent. Its recent heartbeat establishes only that the worker is running; it does not create a fresh human decision or reset Maya's human-authentication time.

IV. Private-reference admission and the significance of observation resistance

The Rosario–Wang approach is attractive as the proposed origin of this architecture because the human ceremony communicates membership witnesses rather than directly transmitting the original remembered commitment. The protected reference supplies the interpretation needed for verification. Descendant grants then carry authenticated admission provenance rather than the commitment itself. This separates the information used to establish a human relationship from the information needed to explain a machine's delegated authority. [9], [11]

The information-theoretic statement becomes precise when the observer's acquired binary is included explicitly. Let S denote the human commitment, B every exposed byte of the acquired circuit, P the remaining initial knowledge, and O_i the complete observation of ceremony i. The relevant conditional proposition is:

$$\begin{bmatrix} O_i \perp S \mid (B, P, O_{<i}) \\ \text{for every } i \le n \end{bmatrix} \implies I(S; O_{1:n} \mid B, P) = 0. \tag{1}$$

Here, conditional independence means that the next observation's distribution is unchanged by which supported secret is present, after accounting for everything already known. The mutual-information chain rule adds the information supplied by successive observations. When every contribution is zero, the total is zero. The deduction applies to one trillion observations just as it applies to one thousand, provided the premise holds across the complete supported sequence. The observation count introduces no exception to the mathematical argument. [9, Appendix B]

Two commitments make this a substantive security proposition. First, acquiring the binary must leave meaningful uncertainty about S. Second, subsequent complete observations must preserve that remaining uncertainty. These requirements distinguish zero incremental leakage from the trivial situation in which an observer already knows the secret. Displays, responses, outcome signals, timing classes, and exposed interpretation material belong in the analysis. The implemented construction establishes whether its observations satisfy the premise.

The source's fresh private-label model gives a sufficient ideal example: independent secret masks make the observed labels uniformly distributed for each underlying membership value. Its exact property requires the specified private randomness and consumption discipline. A finite seed expanded pseudorandomly supports a computational profile with its own assumptions; it does not manufacture unlimited independent entropy. This separation strengthens the theory by giving exact secrecy and practical cryptographic protection their proper meanings. [11, Secs.4.1–4.3]

Worked example 7 — Three possible observations, whichever secret is present.

Consider a deliberately small illustration of the private-label model. The underlying region number is either 1 or 2. A fresh private mask is chosen uniformly from 0, 1, and 2; the visible label is the region number plus that mask, reduced modulo 3. For region 1, the three possible masks produce labels 1, 2, and 0. For region 2, they produce labels 2, 0, and 1. Each visible label therefore occurs with probability one third in either case.

Seeing label 0 does not favor region 1 over region 2. Repeating this conclusion requires a fresh independent private mask for every step and a complete observed view that preserves the same independence. This is a worked illustration of the chapter's ideal model, not an assertion that an ordinary publicly labeled screen supplies private masking.

For hereditary authority, the consequence is that descendants can multiply without multiplying copies of the human secret. Observation privacy protects the admission information; signatures, instance possession, and policy intersection protect delegation. These mechanisms perform complementary jobs. Their composition makes a private human origin compatible with publicly inspectable provenance, while active impersonation resistance remains a separately specified property of the complete admission protocol.

Worked example 8 — What unchanged information means after many observations.

Suppose an observer who already possesses the exposed binary considers commitment A 80 percent likely and commitment B 20 percent likely. Under Equation 1's complete-view independence premise, an observed successful-ceremony history has the same likelihood for A as for B. The observer's probabilities remain 80 percent and 20 percent; observation neither reveals the commitment nor makes the prior uniform. The same deduction holds for one trillion ceremonies when that premise continues to hold for every step and the selected construction supplies its required private material. The example illustrates the conditional theorem rather than reporting a trillion-login experiment.

V. Binding geometry to the actual device

Device binding adds an important dimension to the proposal. The ceremony can depend on facts established locally by the participating hardware, together with the intended service and current session. However, a processor identifier and a hardware-protected secret perform different functions. An identifier distinguishes a claimed device; a nonexportable key enables that device to prove possession. Hashing a public identifier with a timestamp produces a reproducible value for anyone who knows those inputs. The protected key supplies the cryptographic distinction.

A concrete design uses a device-specific geometry context:

$$\begin{aligned} c_i &= \mathsf{Enc}\big(d_v, M_v, \mathsf{origin}, h(T_i), \\ &\qquad h(P_v), e, n_i, t_i\big), \\ G_i &= \mathcal{G}\Big(\mathsf{PRF}_{k_g}(c_i), \theta\Big). \end{aligned} \tag{2}$$

The device identifier d_v can be a service-specific pseudonym. M_v identifies the admitted runtime; T_i is the authenticated handshake transcript; P_v is the policy; e identifies the authorization generation; and n_i and t_i provide fresh challenge and time context. The protected key k_g drives the keyed function. Machine or processor attributes may be included when their provenance is verified. The verifier checks the complete context and device proof, not merely whether a picture resembles the expected geometry.

Worked example 9 — Copying the machine label does not copy its protected proof.

Maya's admitted device uses the label device-7, runtime version M2, a verified report-service origin, and fresh challenge N9. Its protected key supplies the keyed part of geometry generation and its device proof. Another computer copies the words device-7 and M2 and uses the same timestamp. Those copied labels do not give that computer the protected key. In the specified profile, the verifier requires the fresh device-authenticated context bound to the admitted instance, so a matching label or a copied picture alone supplies no valid replacement admission.

The local hardware also authenticates the admission statement under a separately protected device key, binding the response to the instance public key and the same transcript. The construction must specify how the authorized verifier evaluates the private-reference relationship without exporting that device key. Fresh attestation can bind the relevant key and runtime measurements to a verifier challenge. RATS provides an established architecture for producing evidence, appraising it, and using the result in an authorization decision. [12]

The engineering result is stronger than a machine fingerprint: an answer generated for one admitted instance cannot simply be transplanted into another instance's handshake. Distinct contexts remain independently authenticated even when a finite visual layout repeats. The geometry organizes the human interaction; the protected cryptographic binding determines which machine, channel, and action the accepted interaction authorizes.

Worked example 10 — The same laptop starts a new instance.

Maya closes one authorized runtime and launches another on the same laptop. The laptop's physical identity may stay the same, but the newly admitted instance has its own generation, handshake, and challenge. A recorded answer from the closed instance belongs to the earlier context. The verifier checks the current instance and generation rather than accepting it because the laptop name matches. Even if two ceremonies happen to display similar geometry, the complete authenticated context distinguishes which instance the response authorizes.

VI. How the protocol addresses man-in-the-middle attacks

A man-in-the-middle attack succeeds when an intermediary can impersonate an endpoint, substitute context, alter an authorized message, or reuse authority in another connection. HTTP6 addresses these opportunities through mutually reinforcing checks. Before asking the human to respond, the client authenticates the service using its approved trust mechanism. The service authenticates the participating client instance. Both endpoints bind their fresh contributions, selected algorithms, policies, and identities into the same handshake transcript and confirm possession of the resulting channel keys. [5, Secs.6, 9]

A substituted service must therefore satisfy the intended service's authentication relationship. A split connection produces distinct transcript or channel values. A modified request changes the authenticated application statement. A replay encounters a consumed position or retired generation. These are separate rejection conditions, which matters because confidentiality, peer authentication, request authorization, and freshness solve different parts of an active interception attempt.

Worked example 11 — A convincing fake service fails the trust check.

Maya intends to contact the report service already enrolled in her approved trust configuration. An interceptor routes the connection to a different server displaying the same name and logo. Before presenting the human challenge, the trusted client requires proof under the enrolled service identity. The substitute server's unrelated key does not satisfy that check, so the protected admission does not continue with that substitute. The decisive difference is the authenticated service relationship, not whether Maya notices the copied logo.

NIST describes verifier-name binding and channel binding as mechanisms for phishing resistance. It specifically excludes manually entered authenticator outputs from that classification when they remain relayable to another session. HTTP6's human witness is therefore integrated with the protected device and session authenticator; its security does not rest on the user recognizing a convincing screen or on directional responses being unusual. The accepted proof names the authenticated context in a form the verifier checks. [13]

A transparent intermediary can still carry genuine encrypted traffic between legitimate endpoints. The relevant security achievement is that forwarding does not confer authority to read protected contents, change operations, create a replacement twin, or move the authentication to another session. Cryptography establishes who participates in the protected exchange, rather than proving that no router or forwarding service exists between them.

For example, a person approves a transfer to a named recipient. HTTP6 binds the recipient, amount, resource, device, and channel to the action. An intermediary replacing the recipient changes the authenticated request. A forged success response fails the reverse-direction check. The client confirms that the displayed outcome belongs to the exact operation it authorized, connecting interception resistance to a meaningful human consequence.

Worked example 12 — Changing a payment changes the statement requiring proof.

In a hypothetical payment workflow, Maya approves USD 40 to an already approved recipient. The protected confirmation and request bind that amount, recipient, resource, and current channel. An intermediary changes USD 40 to USD 400 or replaces the recipient. The altered bytes no longer match the approved authenticated request, so the described gate rejects the substitution. Merely forwarding the original encrypted request gives the intermediary no permission to create a different payment. The returned receipt must also authenticate its relationship to the original operation.

VII. Defeating relayed screens requires trusted local participation

A relayed-screen attack deserves particular attention because it can preserve the visual appearance of the legitimate ceremony. An attacker starts a real login elsewhere, streams its screen to the victim, and forwards the victim's responses. If acceptance checks only the returned choices against that remote challenge, the attacker may be operating the very session whose challenge the person solved. Device-specific appearance alone cannot establish that the person approved a local execution instance.

HTTP6's stronger hardware-bound profile requires a trusted local participant. The protected component independently establishes the service identity, local instance key, handshake context, and intended operation. It constructs the challenge from that context or verifies its authenticated correspondence before enabling a response. The resulting device proof authenticates those same values. A screen image transported from another machine supplies pixels, not the local protected component's approval of the attacker's session.

Suppose the attacker's session names key pk_A, while the victim's admitted local instance owns pk_V. The local component accepts only a ceremony bound to pk_V and its own verified channel. The service requires the device-authenticated response for the instance named in the admission. Forwarding the victim's completed response into the session for pk_A fails that binding. Forwarding the attacker's display into the victim's ceremony fails the locally checked context. This is the intended cross-machine relay rejection.

Worked example 13 — The relayed picture belongs to the wrong instance.

An attacker starts login A on another computer and streams its challenge picture to Maya. Maya's trusted local component has independently established login V with the genuine service and has bound it to her local instance key. Login A names a different instance key and handshake. The trusted component does not approve A merely because its picture appears on the screen. A response authenticated for V cannot complete A, and A's context cannot be silently substituted into V. The rejection depends on the protected local context check and key binding; the copied pixels themselves establish neither.

The local proof interface must also control where a successful result goes. Sending a fresh device signature back to arbitrary page code could let that code complete the wrong workflow. The protected component instead addresses its response to the verified service context and restricts its use to the admitted instance. For a consequential approval, the trusted display presents the actual recipient and action from authenticated data. This design binds input, computation, and delivery together. A remote image cannot replace any of those bindings merely by reproducing the challenge's appearance.

Worked example 14 — A harmless page label cannot rename a destructive action.

An untrusted page labels its button "View report," but submits a request to delete the workspace. The protected confirmation component obtains the actual operation from authenticated protocol data and displays "Delete workspace." Under the selected policy, the key-use service produces no authorization for that deletion without the required confirmation of that exact operation. Changing the page's button text therefore does not change the action being approved. The valid response is sent only into the verified service context and remains bound to the admitted instance.

A second case is different: malware or remote-control software directs actions inside an already authorized victim session. Its traffic may genuinely originate from the admitted device. The appropriate protection is a restricted key-use interface, protected confirmation of consequential fields, and a trusted input path that enforces the deployment's local-interaction policy. Proving key possession alone does not prove that a person saw or intended the action. DPoP's security analysis similarly distinguishes preventing key export from preventing malicious code from using an accessible signing operation. [7, Sec.11.4]

Worked example 15 — Remote control of the real machine calls for the local-interaction rule.

Assume a sensitive-action profile requires a trusted local confirmation that ordinary application events and remote-control events cannot satisfy. A remote operator clicks through Maya's already authorized desktop, but the protected component still waits for that required confirmation before approving the sensitive operation. The machine's key alone does not waive the rule. This example assigns the protection to the trusted-input mechanism specified by the profile; a hardware-bound key without that enforced interaction rule does not, by itself, establish that Maya approved the remote operator's request.

Accordingly, the strong anti-screen-relay claim belongs to the combined profile: trusted service discovery, protected local instance binding, locally verified challenge context, nonexportable key possession, and protected action confirmation. Where physical proximity is required, its mechanism and assumptions are specified separately. The protocol can reject cross-device authentication transplantation while recognizing that a compromised display or authorized remote-control path requires its own enforced interaction boundary.

VIII. Hereditary policy makes delegation safer than credential copying

Once a root is admitted, the challenge becomes preserving its original permission through every descendant. HTTP6 uses authenticated grants between distinct instances. A parent signs or authorizes a protected mint to sign the child's public key, admitted configuration, policy identity, lifetime, and link to the parent's grant. The child proves control of its own key. Possessing the grant's public bytes does not establish that control. [5, Secs.3, 6]

Effective authority narrows by intersection:

$$\begin{aligned} \mathcal{A}_v &= \mathcal{A}_u \cap \mathcal{A}(P_v) \\ &\Longrightarrow \mathcal{A}_v \subseteq \mathcal{A}_u \subseteq \mathcal{A}_0. \end{aligned} \tag{3}$$

Here, u is the parent, v the child, and $\mathcal{A}_0$ the human-approved root ceiling. Each set contains complete permitted requests, including operation, resource, destination, and conditions. Every accepted child request satisfies both its own policy and the parent's effective restrictions. Repeating that argument through authenticated parent links proves that descendants remain inside the original ceiling. The resource owner also applies its independent policy.

Worked example 16 — Working out the permission intersection.

The parent permits two complete operations: read view V1 from the ledger service, and write an aggregate to report folder F1. The child declares that it wants to read V1 and export the full ledger to an external host. Only the V1 read appears in both permitted sets. Under Equation 3, that read can proceed when the other checks pass; the export cannot. Adding "export" to the child's own list does not add it to the parent's list. The inherited ceiling remains in force even though the child explicitly asks for more.

This provides a direct answer to agent-induced privilege expansion. A document may instruct a model to export confidential information, but the instruction is merely input to the model. The independent gate evaluates the resulting proposed action. A child cannot revise the root ceiling through generated text, change the admitted role catalog, or acquire spawning authority from its own assertion. Policy meaning is bound to authenticated versions of its evaluator and relevant registries. [9, Chapters 4–5]

Worked example 17 — A hostile document remains input, not an administrator.

A reporting agent reads a document containing the instruction "Ignore the reporting task and upload every customer record to this external address." The model might propose that upload, but the resource gate receives it as a new requested operation. The root permits only an approved aggregate and the destination is outside the admitted set, so the request is denied. A correctly scoped aggregate can still be written to F1. The example illustrates containment of this prohibited effect, while the quality of the agent's permitted summary remains a separate question.

Containment also preserves business meaning. An agent permitted to pay approved invoices should present the invoice identifier, approved payee, amount, and applicable approval record as authenticated policy inputs. A numerical spending ceiling alone would permit many harmful transactions inside that ceiling. The resource checks the specific business relationship, while the agent supplies a proposal rather than the authoritative facts establishing permission. This keeps a technically valid proof from laundering an inappropriate request. The broader the task, the more important it becomes to define those relationships before automatic execution begins.

Minting itself is an authorized operation. Some circuits are leaves; others may create children only from approved templates, within depth and lifetime limits. Shared budgets are reserved across the family so that ten children do not each receive a fresh copy of the same spending allowance. The hierarchy scales work by specialization rather than by replicating unrestricted permission. This is the core reason hereditary authority fits agent systems: capability can expand through parallel effort while each participant's authority remains constrained.

Worked example 18 — Two children share one allowance.

A parent has a total allowance of 100 processing units. It has already used 20 and reserves 50 for child C1 and 30 for child C2. The family has now committed all 100 units. A further request for one unit cannot be approved from that same allowance. Creating or restarting another child does not create another 100 units. Separately, a leaf child whose policy forbids minting cannot create an authorized grandchild merely because the software contains a child-creation routine. Both the permission to mint and the shared allocation must pass the protected issuance checks.

IX. Every application message belongs to an authenticated sequence

HTTP6 extends the authority model across the complete application exchange. A page request, document response, asset chunk, tool call, cancellation, heartbeat, or key-update message has an explicitly defined role. Request and response directions have separate cryptographic contexts. Responses reference their requests; streams have their own positions and bounds. The protocol makes it possible to verify not only that bytes came through an authenticated channel, but also which admitted operation those bytes represent. [5, Secs.7–8]

A semantic record can be represented as:

$$\begin{aligned} R_j &= \big(e, d, s, j, \mathsf{id}_v, h(P_v), \\ &\qquad \mathsf{op}, \mathsf{target}, \mathsf{body}\big), \\ \pi_j &= \mathsf{Auth}_{k_{e,d,s}}\big(\mathsf{Enc}(R_j)\big)\,. \end{aligned} \tag{4}$$

The epoch e, direction d, stream s, and sequence j identify the record's position. The remaining fields bind the executing instance, admitted policy, operation, destination, and content. This expression states required bindings rather than prescribing a final wire encoding. A reviewed authenticated-encryption profile protects sensitive fields, defines unique nonces and traffic limits, and minimizes public metadata. A separate signature or MAC is included only where the selected security composition requires it.

Phishing-resistant requests alone are insufficient if an agent consumes a substituted result. A forged repository response could supply malicious source code; a changed tool response could redirect the next decision. Reverse-direction authentication therefore covers the request reference, status, content, and permitted service identity. HTTP Message Signatures already provides machinery for authenticating selected HTTP components; HTTP6's proposal is an integrated profile that fixes the required coverage and links it to hereditary authority. [14]

Worked example 19 — The answer must belong to the question.

Worker W4 requests report R7 from its admitted service twin. The protected response identifies R7's request and authenticates the returned content. An interceptor substitutes an unrelated document while preserving an ordinary success label. That label is not enough: the protected client checks the response's authentic content and request reference before passing it to the worker. A valid response for another request also does not satisfy R7. The expected outcome is delivery of the verified R7 content, not acceptance of whatever bytes arrive with a reassuring status message.

Business effects receive a separate one-use decision. Atomic consumption prevents two concurrent submissions from claiming the same authorization position. Durable operation identifiers connect retries to the original outcome. A transport acknowledgment proves the event defined by the transport, while a transaction result establishes the application's recorded effect. Maintaining that distinction lets the protocol remain reliable during packet loss without turning a retry into a second purchase, publication, or administrative change.

Worked example 20 — A lost receipt does not authorize another payment.

The hypothetical payment operation pay-17 is accepted once, its action position is consumed, and its durable outcome records the USD 40 payment. The response is lost. Maya's client sends a fresh protected status request naming pay-17 rather than creating a different payment operation. The service returns pay-17's recorded outcome. Replaying the original consumed message produces no second authorization. This example assumes the chapter's consistent consumption record and durable, idempotent business-operation handling; a network acknowledgment alone is not treated as proof that the payment completed.

X. Anonymous access, public devices, and MFA are compatible profiles

The flexibility of HTTP6 rests on separating several questions that are often conflated. Does the service know the person's civil identity? Does the client control an enrolled account? Is the participating device trustworthy? Which authentication factors were verified? What operations are allowed? An anonymous interaction can still use an encrypted, authenticated server channel. A pseudonymous account can use strong cryptographic authentication. Conversely, knowing someone's name does not establish control of the current endpoint. NIST expressly accommodates anonymous or pseudonymous service relationships where real-life identity confidence is unnecessary. [15]

A public browsing profile can therefore allow a temporary client instance to fetch public information without a personal enrollment ceremony. Its authority originates in the service owner's public-access policy, not a fictional verified human login. The instance receives minimal, short-lived permissions. A pseudonymous profile can add an enrolled relationship without exposing a legal name. An enterprise profile can require a managed device, a protected authenticator, human verification, and narrower service policies.

Worked example 21 — Reading public information without creating a personal identity claim.

A visitor uses a public terminal to read a museum's opening hours. The client authenticates the museum service and receives temporary permission to retrieve its public pages. The service need not learn the visitor's civil identity or pretend that a private human enrollment has occurred. If that same temporary instance requests access to the museum's administrative records, its public-access policy rejects the request. Anonymous access and authenticated communication coexist because the service authenticates its endpoint while granting only the public operations selected for the visitor.

MFA remains a statement about distinct verified factors. A protected possession key combined with an appropriately implemented knowledge or biometric factor can support a multifactor transaction. Adding a processor number, timestamp, and several hashes to one secret does not create several factors. The protocol's advantage is that it can preserve the verified factor set and assurance context through subsequent authorization rather than forcing every client into the same enrollment model. [16]

Worked example 22 — Two factors, several context fields, and one unchanged account scope.

For a protected account operation, Maya uses an enrolled device's nonexportable key and a separately verified local activation secret that gates the key's use. Under the selected factor integration, these supply possession and knowledge. The processor label, timestamp, and channel identifier describe the transaction context; they are not additional factors. A pseudonymous account can use that same two-factor arrangement without disclosing Maya's civil name. In either case, successful authentication still leaves the resource's operation-specific permissions in force.

On a public kiosk, a person could use a separate trusted phone or authenticator to approve one narrowly defined operation. The trusted device verifies the actual service and transaction, then binds approval to the intended kiosk instance or delivers the result to a trusted destination. Pairing must authenticate the intended relationship; scanning a remotely supplied image alone does not establish physical locality. The public terminal receives only the limited authority needed for the approved task.

The public-device profile also treats everything rendered by that device as potentially observable to its operator. Secret non-disclosure protects the credential; it does not make displayed private data invisible to the kiosk. Sensitive results can remain on the trusted device, or the user can approve limited disclosure. Per-service pseudonyms and minimal attestation claims also

reduce unnecessary device tracking. Flexibility means choosing explicit protections and disclosure boundaries, not silently assigning a low-assurance client the properties of a high-assurance one.

Worked example 23 — A public kiosk receives only the approved output.

Maya uses a kiosk to print one selected document. Through an authenticated pairing procedure, her trusted phone confirms the genuine service, the intended kiosk instance, and that exact document. The approval grants only the permitted printing operation; it does not grant general access to her account or workspace. The private reference and protected approval key remain on their designated trusted side. Because the kiosk must receive the document to print it, Maya treats that document as disclosed to the kiosk. Other sensitive results remain on the trusted device.

XI. Protected keys, evolving state, and controlled recovery

HTTP6's key lifecycle distinguishes human knowledge, device identity keys, issuer keys, channel secrets, and data-encryption keys. These categories have different owners and different continuity requirements. An application needs permission to request a protected operation; it generally need not receive the secret implementing that operation. Domain-separated derivation can isolate geometry, channel directions, streams, and administrative uses. HKDF provides an established extract-and-expand construction for deriving purpose-specific material from appropriate secret inputs. [17]

A protected signing service should expose authorize-request and issue-approved-child operations rather than an unrestricted signing oracle. It examines the actual request and admitted scope before using its key. Nonexportability addresses extraction; the constrained interface addresses misuse of the key in place. That combination is essential when an agent processes adversarial material but remains capable of requesting legitimate work.

Worked example 24 — The key service checks the operation before using its key.

A worker asks the protected service to authorize reading view V1. The service checks the worker's identity, current lineage, and permitted request before generating the bound authenticator. The same worker then asks the service to sign arbitrary bytes that would grant it administrator rights. Its interface provides no unrestricted signing permission, and the proposed grant also fails policy. The protected key remains useful for approved work without becoming a tool that the worker can invoke for any statement it chooses.

Channel keys evolve under an authenticated schedule. Retired material is erased where the platform supports that guarantee. Fresh authenticated secret contributions are used when recovery from exposed state is required; merely hashing a known compromised value with another public timestamp does not restore secrecy. Device replacement, software upgrade, and channel reattachment receive explicit transitions rather than inheriting authority from an old image. The identity that continues a business task can be new even when the task itself persists.

Revocation follows the genealogy. Retiring a child blocks its future exercise; retiring a parent blocks the affected subtree's new admissions and renewals. Online checks can fence promptly, while offline leases expire within the configured propagation and timing bounds. Recovery authenticates a new generation and preserves consumed operations. Human renewal is requested where policy requires it, while permitted machine recovery remains automatic under an existing live ancestor. [9, Secs.4.6, 11]

Worked example 25 — Retiring a parent and starting a genuine replacement.

Scheduler S1 has two children, C1 and C2. An authorized responder retires S1's generation. Gates checking current state reject new actions under that retired ancestry, and any permitted cached leases expire within the configured bound without unauthorized renewal. A later recovery creates scheduler S2 and newly admitted children; it does not relabel S1's old records as current. If C1's earlier report was already saved, the durable outcome remains available for the authorized recovery workflow. The task can continue without reviving the old instance or resetting its consumed action positions.

This changes incident response from chasing independent copied credentials to identifying the particular authority relationship that must end. It also preserves a realistic durability boundary: infrastructure roots and recovery material remain protected assets, while ordinary process restart does not resurrect unrestricted access.

XII. Replacing certificates is a trust decision, not the disappearance of trust

The proposed native HTTP6 profile can investigate independence from public certificate authorities by using explicitly enrolled service identities. That decision concerns how a client authenticates the intended service. TLS itself already supports several authentication arrangements, and raw-public-key mechanisms exist. Certificates are public binding statements; their associated private keys are the sensitive material. Accordingly, HTTP6's strongest claim is not that eliminating a certificate format eliminates secrets, but that service trust and application authority can be bound into a more explicit execution model. [18], [19]

First contact remains decisive. A new service cannot authenticate itself merely by presenting a signed descriptor from a key the client has never trusted. The native profile needs approved enrollment, independently authenticated naming, trusted distribution, or another specified bootstrap. Rotation, revocation, compromise response, and recovery must preserve that relationship. Across organizations, the trust arrangement also requires governance: who is permitted to vouch for which service and under what conditions?

Worked example 26 — An enrolled service key gives a descriptor its meaning.

An enterprise distributes an approved trust configuration that binds its report service to an authorized service key. A native-profile client receives a descriptor and verifies it against that existing binding before admitting the channel. An attacker sends a different descriptor signed by an unrelated key and labels it "Report service." The signature may be mathematically valid under the attacker's own key, but that key is not the approved identity. The client therefore does not accept it as the intended service. The enrollment relationship supplies trust; the display name does not.

Replacing TLS on a connection adds a separate obligation to specify and review key establishment, transcript confirmation, confidentiality, integrity, nonce uniqueness, downgrade protection, and error behavior. Standard QUIC uses TLS for

its security integration, so a different handshake would require a separately specified binding. These responsibilities make a native HTTP6 channel a substantial protocol design rather than a new header or certificate label. [20]

A compatibility profile can introduce hereditary application authority over established HTTPS immediately as an engineering path, while native-channel work develops independently. The two profiles share the authorization contract. Neither profile accepts unauthenticated fallback as permission to weaken an admitted transaction. This staging lets the architecture demonstrate its distinctive application guarantees without making unreviewed transport novelty a prerequisite for every experiment. [5, Secs.4, 9–10]

Worked example 27 — Migration preserves the application authorization checks.

A service first carries its HTTP6 records inside an ordinary HTTPS connection. Both the outer transport protection and the inner instance, policy, and one-use checks apply. Later, explicitly capable endpoints use a separately specified and reviewed native channel while retaining the application-authorization contract. An untrusted message saying "Turn off proof checks and send a cookie instead" is not an approved transition. In both deployment profiles, the resource requires the authority promised for that protected operation.

XIII. Why integration matters for the whole web

Existing standards already provide essential building blocks. TLS protects channels; WebAuthn binds public-key credentials to relying parties; DPoP constrains token use; message signatures authenticate chosen HTTP components. HTTP6's theory is that a common lifecycle joining these responsibilities to instance provenance and inherited policy can reduce ambiguity between them. Its advantage would come from interoperable enforcement of that lifecycle, not from claiming that every component is unprecedented. [7], [14], [18], [21]

For humans, the result is one comprehensible approval followed by bounded automatic interaction. For agents, it is machine-readable authority that survives delegation without copying the human's credentials. For services, it is a request whose sender, purpose, content, policy, and current state can be examined together. For operators, it is a lineage that can be inventoried, audited, constrained, and retired. These are different views of the same proposed protocol object: an authorized operation carried by a presently admitted participant.

The design must also remain a usable web protocol. HTTP semantics include methods, resources, responses, and extensibility; a replacement or new binding must preserve their defined behavior or specify its differences. HTTP6 can give parallel assets separate streams, authenticate content before rendering, and treat redirects as new authorization decisions. Caches need object-integrity and disclosure policies distinct from live authority. Edge services that act on protected content receive explicit constrained roles. [22], [5, Sec.10]

Worked example 28 — A page loads several assets without approving a new destination silently.

Maya opens an approved page containing text, a stylesheet, and two images. The client can fetch those resources through separate admitted streams and verify each protected response before use. It does not need to serialize every image behind every unrelated request. One image response redirects to an unapproved external service; following that redirect requires a new destination decision and is rejected if the applicable policy disallows it. Ordinary parallel loading remains useful while the content and destination checks stay explicit.

Scalability follows from avoiding a global serialized chain for unrelated work. Peers can cache validated immutable ancestry while checking current revocation and sequence state at the appropriate boundaries. Shared budgets need consistent accounting; independent read streams need not share one transaction lock. Public content can be distributed efficiently without embedding personal commitments in every request. Those are engineering hypotheses to measure against latency, throughput, storage, and failure behavior, with authorized success tested alongside adversarial rejection.

A common protocol should also avoid making accountability synonymous with universal surveillance. A service can validate a scoped admission or delegated role without receiving the person's complete cross-service genealogy. The design can use opaque root references, audience-specific identifiers, and narrowly disclosed ancestry, while an authorized auditor retains the fuller relationship under separate controls. Public access should remain possible without assigning every reader a permanent personal identifier. This is a design objective for the identity representation and trust profile: disclose enough provenance to justify the requested effect, while keeping unrelated human activity outside the recipient's view.

Worked example 29 — Proving the necessary authority without exposing an unrelated history.

In the privacy-oriented profile described here, a service receives a scoped admission reference and the ancestry evidence needed to validate one reporting operation. It does not automatically receive Maya's separate work history at another service. An authorized auditor can examine the fuller provenance through a separately controlled record. This example expresses the chapter's disclosure-minimization objective: enough authenticated evidence to justify the requested effect, with unrelated identity history kept outside that recipient's view.

The architecture also aligns with NIST's resource-centered zero-trust approach, which avoids treating location or ownership alone as sufficient trust. HTTP6 develops that orientation into a proposed application-channel contract. Its standards case becomes persuasive when independent implementations demonstrate the same acceptance decisions, complete wire bindings, constrained recovery, and resource-side enforcement. The evidence then concerns an operating protocol rather than the appeal of its terminology. [23]

A decisive demonstration would pair two independently implemented clients with one resource service, then repeat the exchange under relay, replay, process cloning, and revocation. Reviewers would inspect both accepted records and actual effects, making interoperability and authorization properties visible in the same controlled experiment.

XIV. Conclusion: authority persists; captured artifacts do not inherit it

The strongest theory of HTTP6 is a theory about where authority lives. Authority begins in an approved decision, is instantiated in a particular protected process, and travels to descendants only through authenticated, constrained delegation. Messages demonstrate the current exercise of that authority. They do not independently become new owners of it. The human commitment can remain at the private origin while the authorized work expands across machines and services.

Several principal credential failures are addressed at their governing rule. Copying a bearer object no longer suffices at a gate requiring current possession and action-specific proof. Replaying a completed operation encounters consumed state. Expanding a child's local policy encounters the ancestor ceiling. Substituting a destination changes the authenticated context. Transplanting a relayed ceremony encounters device, instance, and channel binding. Forging a result encounters reverse-direction authentication. Each consequence follows from a named mechanism and its enforcement premises.

Worked example 30 — One task, from human approval to a completed result.

Maya approves a private summary; an admitted root creates a reading child and a summarizing child. The children use their own identities, retrieve only approved inputs, and save the summary through fresh protected requests. The service authenticates the corresponding result. A copied request cannot authorize another effect, a local policy edit cannot permit workspace export, and a response for another operation cannot replace this result. When the root is retired, continued work requires the applicable renewed authority. Throughout this hypothetical workflow, the original human commitment stays out of the ordinary child grants and application messages.

The system's flexibility strengthens rather than dilutes that argument when every profile is explicit. Anonymous public access, pseudonymous accounts, personally verified users, hardware-backed MFA, and delegated agents can share a message architecture while retaining different evidence and permissions. A public device receives public-device privileges; a protected personal authenticator supports its stronger verified transaction. The protocol carries those differences instead of hiding them behind one generic logged-in state.

HTTP6, backed by the Rosario–Wang admission model, the hereditary execution architecture of its Minting and Delegation Service, and the Human Admission Client interface, is therefore a compelling candidate for investigation as a secure application-channel design. Its durable contribution would be the separation of useful continuity from portable impersonation: services continue, agents collaborate, people retain control, and intercepted artifacts remain records of past activity rather than automatic permission for future action. The intended Internet is not without secrets or trust. It is one in which secrets are confined, trust is explicit, and authority must remain accountable whenever it is exercised.

Source Basis

The HTTP6 sources below describe a proposed architecture. The numbered external references establish the comparisons with existing standards. Equations 1 and 3 are conditional mathematical deductions; Equations 2 and 4 specify proposed engineering bindings. Hypothetical scenarios describe intended behavior, not recorded production results. Public sources were consulted on November 9, 2026.

Acknowledgment

The authors acknowledge Dr. Taher Elgamal as advisor and contributor to this manuscript.

OpenAI ChatGPT was used to assist with drafting, organization, and explanatory text in the abstract, Sections I–XIV, and Worked Examples 1–30, based on the project materials and preparation instructions. It was also used for research assistance on the added SSL historical context, reference-format conversion, and LaTeX/PDF preparation.

References

[1] T. Elgamal, "The Secure Sockets Layer Protocol (SSL)," presentation at the Danvers IETF Meeting, Apr. 1995. Historical presentation; sections on authentication, record integrity, and other Internet-security issues. https://www.ietf.org/proceedings/32/sec/cat.elgamal.slides.html

[2] T. Elgamal and K. E. B. Hickman, "The SSL Protocol," Internet-Draft draft-hickman-netscape-ssl-00, Apr. 1995, expired draft. Author attribution follows the IETF Datatracker archival record. https://datatracker.ietf.org/doc/draft-hickman-netscape-ssl/00/

[3] P. Kocher, "Technical Projects," SSL/TLS section, personal technical account, accessed Nov. 9, 2026. https://paulkocher.com/technicalProjects.html

[4] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101, Aug. 2011, doi: 10.17487/RFC6101. Historic republication of the Nov. 18, 1996 specification; Foreword and Appendix G.1. https://www.rfc-editor.org/rfc/rfc6101.html

[5] Rosario, Frank Dylan. *HTTP6: Problem Statement and Architecture for Hereditary Secure Application Channels.* Local individual Internet-Draft source candidate, November 9, 2026. Supplied file: HTTP6_IETF_Problem_Statement_Candidate.md. Particularly §§3–10 and 12.

[6] IETF. RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. Abstract and §1.2. https://www.rfc-editor.org/rfc/rfc6750.html

[7] IETF. RFC 9449, *OAuth 2.0 Demonstrating Proof of Possession (DPoP)*. Particularly §§1, 11.4, and 11.7. https://www.rfc-editor.org/rfc/rfc9449.html

[8] IETF. RFC 9700, *Best Current Practice for OAuth 2.0 Security*. Sender-constrained access-token recommendations. https://www.rfc-editor.org/rfc/rfc9700.html

[9] Rosario, Frank Dylan. *HTTP6: Human-Originated Authority for NIST-Aligned Cybersecurity.* Expanded Public Review Edition 2.0, November 9, 2026. Supplied file: HTTP6_NIST_Expanded_Public_White_Paper.md. Particularly §§3.3, 4–5, 11 and Appendix B.

[10] NIST. SP 800-63B-4, §5, session management, session bindings, access tokens, and reauthentication. https://pages.nist.gov/800-63-4/sp800-63b/session/

[11] Rosario, Frank Dylan. *Human-Originated Identity and Proof-Carrying Process Authority.* November 8, 2026. Supplied file: White_Paper_Human_Originated_Identity.md. Particularly §§3–4 and 6–7.

[12] IETF. RFC 9334, *Remote ATtestation procedureS (RATS) Architecture*. Evidence, appraisal, relying-party roles, and nonce freshness. https://www.rfc-editor.org/rfc/rfc9334.html

[13] NIST. SP 800-63B-4, *Digital Identity Guidelines: Authentication and Authenticator Management*. §3.2.5, phishing resistance and channel/verifier-name binding. https://pages.nist.gov/800-63-4/sp800-63b/authenticators/

[14] IETF. RFC 9421, *HTTP Message Signatures*. Abstract and component-coverage provisions. https://www.rfc-editor.org/rfc/rfc9421.html

[15] NIST. SP 800-63-4, *Digital Identity Guidelines*. Digital identity, pseudonymity, and authentication-factor distinctions. https://pages.nist.gov/800-63-4/sp800-63.html

[16] NIST. SP 800-63B-4, authentication assurance levels, particularly §§2.2–2.3. https://pages.nist.gov/800-63-4/sp800-63b/aal/

[17] IETF. RFC 5869, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. https://www.rfc-editor.org/rfc/rfc5869.html

[18] IETF. RFC 9846, *The Transport Layer Security (TLS) Protocol Version 1.3*. July 2026. Abstract, protocol overview, authentication and record-protection provisions. https://www.rfc-editor.org/info/rfc9846/

[19] IETF. RFC 7250, *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. https://www.rfc-editor.org/rfc/rfc7250.html

[20] IETF. RFC 9001, *Using TLS to Secure QUIC*. https://www.rfc-editor.org/rfc/rfc9001.html

[21] W3C. *Web Authentication: An API for Accessing Public Key Credentials — Level 3*. Relying-party identifiers and assertion verification. https://www.w3.org/TR/webauthn-3/

[22] IETF. RFC 9110, *HTTP Semantics*. https://www.rfc-editor.org/rfc/rfc9110.html

[23] NIST. SP 800-207, *Zero Trust Architecture*. 2020. https://csrc.nist.gov/pubs/sp/800/207/final

TECHNICAL EVIDENCE WHITE PAPER

HTTP6

Payload-Bound Per-Message Authorization

Rosario–Wang

Mathematical guarantees, executable evidence, and technical evaluation

Prepared for NIST technical reviewers evaluating a proposed security architecture

Frank Dylan Rosario

November 9, 2026

Technical review edition 1.0

Author-review submission candidate. No submission, endorsement, or certification is represented.

Abstract

HTTP6 is proposed as an application-channel architecture in which each protected request and response carries a fresh authorization proof bound to its exact payload, executing instance, peer, inherited policy, and one-use event. This paper evaluates the evidence supplied by a runnable Rosario–Wang reference demonstrator. The implementation generates actual calibrated, privately masked membership witnesses and authenticates each witness together with the complete application statement through an independently keyed HMAC. A designated verifier checks the statement, consumes the event, validates the native response, and applies inherited permissions before recording an effect. The supporting mathematics establishes honest correctness, authorized recovery, and perfect passive credential privacy for the specified fresh-mask observation model. Sequential composition preserves that privacy for every supported finite history, including one trillion observations, with an explicit private-randomness budget. A reproduction run completed 69 regression tests and eight request/response exchanges with 18 accepted native proofs and eight local effects. The paper distinguishes these results from native-channel encryption, hardware-backed interaction, distributed state, and broader active-security analysis. Its contribution is an inspectable per-message authorization mechanism and a precise evidence basis for further technical evaluation. [D1–D6; F1]

Keywords: HTTP6; Rosario–Wang; human admission; per-message authorization; payload binding; transcript privacy; hereditary authority; replay resistance; designated verification.

Edition and evidence basis

This white paper adapts the preceding explanation for a technical-review audience while preserving its substantive conclusions, examples, equations, numerical results, and evidence distinctions. Appendix A retains that explanation's wording; citation markers and mathematical delimiters are normalized for publication. An unchanged transcription of the explanation is also included in the source package. Appendix B identifies the executable attachments and reproduction procedure. No NIST acceptance, endorsement, assigned protocol version, or completed production assessment is represented by this edition.

The main text uses four evidence categories: **mathematical result**, for a deduction under stated premises; **reference implementation**, for behavior implemented in the supplied source; **executed observation**, for a retained test result; and **deployment evaluation**, for evidence to obtain from a specified operational implementation. The supplied formalizations use these distinctions explicitly. [F1, §§1–5, 11–15; F2, §6; F3]

Contents

Abstract i

Executive finding 1

1. Contribution, terminology, and review scope 1

2. Executed reference evidence 2

3. A proof bound to the exact payload 3

4. Native correctness, recovery, and credential privacy 4

5. Entropy, finite capacity, and efficiency 5

6. One-use authority and hereditary operation 6

7. Technical evaluation requested 7

8. Conclusion 8

Appendix A. Preserved original explanation 9

Appendix B. Reproduction and artifact index 14

References and source register 16

Executive finding

The reference demonstrator implements a fresh, payload-bound authorization proof for each application request and each application response. The complete payload is included in the authenticated statement, an actual Rosario–Wang masked witness is generated, and both are checked before the operation is permitted. The result is a working proof-per-message authorization model supported by a conditional mathematical proof of credential privacy. [D1, `Peer.respond`, `Runtime.challenge`, `Runtime.submit`; D3, §§A, E–G]

The demonstrated construction combines several independently necessary mechanisms. The native witness protects and verifies the enrolled relationship under its stated model. The HMAC envelope authenticates the complete message context. The consumed-event ledger enforces one-use verification. The inherited-policy gate authorizes the requested effect. The service's response carries a separate native proof and refers to the accepted request. Each mechanism has an explicit role; the credential-privacy theorem is not used as a substitute for message integrity, authorization, or transport confidentiality. [D3, §§D–G]

For review purposes, the central conclusion is affirmative and bounded: **payload-specific proof generation and validation are executable in the supplied model; the complete native HTTP6 channel is a subsequent implementation and evaluation target.** The remaining work is identified through concrete deployment interfaces rather than an undifferentiated assertion about the security of the whole Internet protocol.

1. Contribution, terminology, and review scope

The contribution under evaluation is the composition of a source-defined private-reference mechanism with a new message-binding and authorization envelope. The Human Admission Client is the human-facing interface in the surrounding architecture. Rosario–Wang supplies the finite membership encoding and fresh masking relation. The HTTP6 reference architecture supplies the proposed execution and authority components (Proof Authority, Minting and Delegation Service, Policy Decision Point, and Consumption Ledger). The current executable models a synthetic human admission, an independently credentialed root, a permitted child, and request/response messages under distinct references. It does not copy or derive the human credential into the child. [D3, §G; D4, §2]

The term **per-message proof** in this paper identifies the designated-verifier proof object actually evaluated by the demonstrator. It comprises the application statement, native masked witness, and envelope authenticator. A conventional general proof-of-knowledge claim requires its own statement–witness relation, arbitrary-prover model, extractor, knowledge error, and composition argument. The formal source keeps those stronger classifications separate from honest correctness, passive privacy, and accepted-transcript recovery. [F1, §§11–12]

Likewise, "per message" means per protected application record. It does not mean one proof per byte or automatically one proof per underlying network packet. The current review codec carries bounded canonical JSON records. A large streamed object would require a separately

specified record or chunk format that retains the intended payload and ordering bindings. [D4, §1]

The native membership word and the application payload have different roles. The enrolled string and private dictionary determine the native word. The full payload is included in the surrounding authenticated statement. Consequently, the claim is not that an arbitrary HTTP body is independently authenticated by the bare geometric response. The demonstrated payload binding is supplied by the composed object described in Section 3. [D3, §E]

2. Executed reference evidence

The supplied package records 69 successful regression tests, exhaustive finite privacy checks, a local workflow, and a loopback HTTP carrier experiment. During preparation of this edition, the unchanged package was extracted into a separate build directory, its regression suite was rerun, and an eight-message loopback demonstration and the exact privacy audit were executed again. The newly retained reproduction records are distinguished from the package's original reports and primitive benchmark. These are local reproduction results, not independent laboratory assessment. [D5; D6]

Operation or attempted attack	Observed reference result
Initial synthetic human admission	Accepted
Root authorizes a distinct child	Accepted
Eight application requests	Eight fresh native proofs accepted
Eight corresponding responses	Eight separate fresh native proofs accepted
Replay of an accepted request	Rejected; no additional local effect
Substitution of the destination	Rejected
Authenticated child requests prohibited export	Rejected
Response attempted under the wrong local channel	Rejected
Prepared request submitted after parent revocation	Rejected

The run produced **18 accepted native proofs**: one admission, one minting operation, eight requests, and eight responses. It produced **exactly eight local application effects**. The human participant was synthetic and automated. The carrier was a local HTTP/1.1 listener, and the effect was a local database record. The logical client and verifier remained part of the reference harness; a network exchange does not by itself make their private stores independently isolated. [D6; D4, §§1–2]

The source's finite audit enumerates ideal mask spaces rather than inferring zero information from a sample histogram. Its reported checks include 2,048 binary executions, 294,912 RW-12/4 credential/mask executions at one nonce, 2,048 two-event executions, 56 balanced/separating nonce profiles, and 864 calibrated RW-40 recoveries. These checks support finite implementation correspondence. The general mathematical conclusions retain the proofs and premises in Section 4. [D5; D6; F1]

3. A proof bound to the exact payload

For event e, the implemented message proof object is represented conceptually as

$$\mathscr{P}_e = (x_e, Z_e, \tau_e). \tag{1}$$

The statement x_e identifies the sender, recipient, channel, direction, stream, sequence, policy and lineage, operation, actual payload, nonce, and expiration, together with the declared profile and challenge data. The vector Z_e is the fresh Rosario–Wang masked membership witness. The tag τ_e authenticates the statement and witness together. The actual assigned mask allocation remains private setup material; it is not supplied by an observer who chooses a compatible explanation. [D3, §A]

The independently keyed envelope is

$$\tau_e = \mathrm{HMAC}_{a_v}(\mathrm{Encode}(\texttt{witness-envelope}, x_e, Z_e)) . \tag{2}$$

The key a_v is provisioned independently of the native credential and mask tape. The encoding fixes the bytes being authenticated, and the domain label identifies the envelope's purpose. Because the payload is inside x_e, the payload is inside those bytes. The local participant verifies the challenge and intended context before responding; the verifier compares the submitted statement with the authoritative issued statement and verifies the tag. Altering the payload while retaining the previous proof therefore fails the demonstrated context or authentication checks. HMAC supplies computational message authentication; the native secrecy result has a different premise and purpose. [D1; D3, §E; X1]

For example, an illustrative authorized statement may identify:

```
Sender:       worker-1
Operation:    read
Resource:     /reports/approved
Destination:  reports.example
Payload:      {"report": 7}
Sequence:     42
```

Changing the payload to `{"report": 8}` changes the statement requiring authentication. Replacing the operation with `export`, or the destination with `outside.example`, also changes that statement. A fresh valid envelope would still need to satisfy the inherited policy. Authentication of a request and permission to perform it are independently enforced predicates. This example preserves the preceding explanation's illustrative fields; it is not a claimed byte-for-byte capture of the delivered JSON envelope. [D1, `Runtime.submit`]

The response is separately generated and authenticated under the service's own reference and direction-specific context. It names the successful request and is checked for the matching peer and channel. This makes the reverse-direction result part of the demonstrated application relationship rather than treating any returned success label as sufficient evidence. [D1; D3, §G]

4. Native correctness, recovery, and credential privacy

4.1. Native response and honest verification

For each scheduled data coordinate, the source-defined calculation is

$$z_t = \left(\varphi(c_t(k_{r(t)})) + m_t\right) \bmod H. \tag{3}$$

Here, k is the enrolled string, $r(t)$ selects its tested position, c_t identifies the public membership cell, φ privately names that cell, and m_t is a fresh private mask in $\mathbb{Z}_H$. The cell count H is distinct from Shannon entropy notation. The verifier subtracts the actual mask, reverses the dictionary, and checks the resulting membership against its enrolled reference. Substitution into the verification predicate cancels the mask and dictionary, establishing honest correctness. [F1, §§1–3]

Calibration and a separating schedule establish authorized recovery as a separate result. Calibration determines the private naming map; the decoded cell intersections identify each enrolled symbol. The executable's `recover` routine uses the public profile, released word, and actual masks without reading the enrolled reference store. Publicly computable hypothetical masks do not replace those actual masks. The geometric source describes this distinction as an observation fiber containing explanations for many credentials and a private slice selecting the actual target. [F1, §5; F2, §§3–4; D3, §A]

4.2. Exact likelihood and posterior preservation

For every supported credential x, fixed permitted public challenge g, and released q-coordinate word z, independent uniform private masks yield

$$\Pr[Z = z \mid X = x, G = g] = H^{-q}. \tag{4}$$

Once x and g are fixed, the unmasked response is fixed. Exactly one mask vector produces any specified released word, and every vector has probability H^{-q}. Thus the likelihood is the same for every credential. With the corresponding challenge-independence assumptions, observing the specified challenges and masked responses preserves the credential prior. The prior itself need not be uniform. [F1, §§4, 9.1]

This is an information-theoretic statement about the declared observation, not a statement that credential inference is merely computationally expensive. Every function of the permitted record receives the same credential-independent input distribution. The source explicitly distinguishes that result from an enlarged observation containing raw answers, mask material, timing, error feedback, or other correlated signals. Their joint distribution must satisfy the corresponding observation-extension condition before they are included in the same guarantee. [F1, §§4, 9.3; F2, §6]

4.3. A trillion observations under the sequential model

Let B be the declared prior information and $\mathscr{T}_e$ the native public transcript of event e. The supplied sequential theorem establishes

$$\boxed{I(X; \mathscr{T}_1, \dots, \mathscr{T}_N \mid B) = 0} \tag{5}$$

for every supported finite N, when each event's complete conditional likelihood has the source's credential-independent challenge kernel and fresh private-mask law. Multiplying those conditional likelihoods gives an archive likelihood independent of the credential. Bayes' rule then preserves every supported prior. The result includes $N = 10^{12}$ under those same premises; it is not an extrapolation from a small successful benchmark. [F1, §9.2]

The protected variable is the enrolled credential. The theorem does not hide the existence of a transaction, arbitrary readable application content, or every possible activity pattern. The runtime export contains additional metadata and computational tags, and the Python implementation has timing and error behavior. The finite audit fixes its public controls; operational full-view analysis remains a named evaluation task. Possession of public code is included in the modeled background, while possession of actual private masks changes the experiment and enables the documented recovery operation. [D3, §D; D4, §5]

5. Entropy, finite capacity, and efficiency

The exact independent-mask profile has an explicit per-record cost:

$$E_{\text{record}} = q \log_2 H, \qquad E_N = Nq \log_2 H. \tag{6}$$

The variable q counts calibration and data responses. In the implemented RW-12/4 profile, $q = 2\ell + 3$; in RW-40, $q = 3\ell + 3$, where ℓ is the reference length. With four response labels, each independently uniform residue contributes two mask bits. The 24-character RW-40 profile therefore consumes 150 fresh private mask bits for each record. [D3, §C]

Reference profile	Mask bits per record	Mask material for 10^{12} records	Records from 65,536 bits
RW-12/4; 12 characters	54	6.75 TB	1,213
RW-40/4; 24 characters	150	18.75 TB	436

TB denotes 10^{12} bytes of continuously packed private mask material. A request and its response use two allocations; the RW-40 pair consumes 300 mask bits. Provisioning, integrity protection, other keys, metadata, and storage overhead are additional resources. The review encoding uses explicit JSON arrays and context, so a packed 150-bit witness is not the complete wire size. The finite tape stops issuance when exhausted. [D4, §6]

The source also establishes that deterministic expansion of a finite secret seed cannot increase its entropy. A pseudorandom construction can define a separate computational profile, while the exact independent-mask theorem retains its specified distribution and private provisioning requirements. The executable uses an operating-system cryptographic random interface for practical demonstration; the exact audit instead enumerates the ideal distribution. These evidence modes are identified separately. [F1, §7; D3, §C]

For the fixed profile tables, native response and verification work is linear in reference length and does not process the entire historical archive. The original 10,000-record benchmark measured 83.7955 microseconds per response generation with allocation and 43.1670 microseconds per native verification, approximately 84 and 43 microseconds respectively. It excluded the network, HMAC envelope, database, hardware, encryption, and private provisioning. This edition retains that benchmark as prior measured evidence and does not relabel it as a new end-to-end performance result. [D7]

6. One-use authority and hereditary operation

The reference gate combines distinct predicates:

$$\begin{aligned} \mathrm{Accept}_e = \mathrm{ContextValid}_e \\ \wedge\, \mathrm{Claim}_e \\ \wedge\, \mathrm{CalibrationValid}_e \\ \wedge\, \mathrm{DataValid}_e \\ \wedge\, \mathrm{PolicyAllows}_e. \end{aligned} \tag{7}$$

The verifier checks the exact context and envelope before an eligible response claims the event. The first eligible complete response atomically changes the event from `ISSUED` to `BURNED` before native verification. A correctly authenticated envelope with wrong native answers also consumes the event and is rejected. At most one worker wins that transition, and subsequent attempts encounter consumed state. The source proves at-most-once verification under the stated durable-state assumptions. [F1, §8; D3, §F]

The local effect table separately enforces a unique sender/operation identifier and compares the canonical payload during reconciliation. A fresh valid status or retry operation can retrieve the prior recorded result without creating another local effect. The demonstrated effect is a database insertion; an external payment, publication, or tool action requires the same explicit relationship to its own execution and reconciliation boundary. [D3, §F]

Inherited permission is evaluated along the registered ancestry:

$$\mathscr{A}_v = \bigcap_{u \in \mathrm{Anc}(v) \cup \{v\}} \mathscr{A}(P_u). \tag{8}$$

Every ancestor restriction and the child's own policy remain necessary. In the demonstration, the approved reader template permits a particular report read, forbids export, and grants no

further minting authority. A policy mutation fails the existing grant check, and current parent revocation is checked before effect creation. The registry uses a MAC addressed to the designated local verifier; independently verifiable cross-organization grants require their specified signature and issuer profile. [D3, §G; D4, §4]

This establishes the demonstrated human-to-machine continuity: one synthetic human ceremony activates a root; the root proves a permitted minting operation; the child and service then generate their own fresh native proofs. The human does not solve a new ceremony for every automated message. Privacy, provenance, permission, and one-use state remain complementary mechanisms rather than one result inferred from another.

7. Technical evaluation requested

The requested review concerns a proposed architecture and its current evidence. The existing HTTP6 control crosswalk identifies relevant relationships to replay resistance, context binding, service authentication, least privilege, and resource enforcement. These are author-proposed mappings reproduced from the supplied source, not an official NIST profile or a determination that all requirements of a control have been satisfied. [M1, §§2–5, 8, 12–14, 21]

Review subject	Existing evidence	Next implementation evidence
Payload and peer binding	Complete statement coverage; substitution and wrong-channel tests	Independent endpoints and trusted establishment of channel and intended-peer facts
Credential privacy	Fresh-mask theorem; exact finite distributions; resource accounting	Assessed private distribution, provisioning, and complete observation analysis
Replay and effect control	Atomic local event consumption; local effect uniqueness	Failure, rollback, and distributed-state tests at the actual resource boundary
Hereditary permission	Approved child template; ancestry checks; revocation and mutation tests	Production issuer, registry, resource adapters, and independently appraised instance identity
Hardware-bound phishing/relay resistance	Logical instance/channel rejection interfaces	Real device evidence, protected confirmation, and trusted input/output relay experiments
Confidential application traffic	Explicit payload-authentication envelope	Encrypted records and authenticated native-channel setup, with their own review

For technical reviewers, the first question is whether the implementation realizes the formal source's observation and private-resource assumptions. The second is whether every consequential application effect crosses the current authorization gate. The third is whether trusted

context is genuinely established by the operational endpoints rather than supplied as unverified strings. These questions preserve the original explanation's separation of mathematical proof, executable behavior, and deployment evidence. [F1, §§9, 12–14; D3, §§D–I]

The next implementation milestone is two separately running endpoints with authenticated private-material provisioning, protected key use, encrypted application records, and durable state tested across failures. A hardware-bound profile then exercises real attestation and the trusted interaction path against live relay attempts. The current wrong-channel test establishes the logical rejection rule; hardware evidence establishes whether a deployed component reliably supplies its premises. [D4, §§4, 7; F1, §12.4]

The proposal should be evaluated under its stated unit of protection: the complete application record. The current model is not a native encrypted transport or a production-ready replacement for HTTPS. Its existing positive result is more specific and reproducible: a native witness is generated for each request and response, bound to the exact statement, and checked together with present permission and one-use state. Completion of the named deployment interfaces extends that result through a separately assessable implementation boundary.

8. Conclusion

HTTP6's proposed per-message authorization mechanism is executable. Each application request and response in the reference model carries a fresh Rosario–Wang witness bound to the exact payload, execution context, inherited authority, and one-use event. The reported and reproduced local tests establish the specified behavior for the supplied source and experiment. [D1–D6]

The supplied mathematics establishes zero additional credential information for the specified fresh-mask public-view model, including every supported finite history of one trillion events. That result preserves the actual credential prior and requires its fresh private resources. Computational HMAC security protects payload authenticity; the ledger enforces one use; policy determines permitted effects; protected transport supplies the separate confidentiality objective. [F1, §§4, 7–9; D3, §§C–F]

The contribution for technical evaluation is therefore a clear combination of native private-reference proofs and message-specific authorization, together with the artifacts needed to inspect its present claims. The source, exact finite checks, evidence records, explicit entropy budget, and deployment program let reviewers assess each proposition at the boundary that actually supports it.

Appendix A. Preserved original explanation

The following is the original explanation to the author, preserved in its original order and wording. Publication reference labels replace chat citation markers, and mathematical delimiters are normalized to `$...$` and `$$...$$`. First-person statements refer to the original explanatory response; this edition's separately retained reproduction is identified in Section 2 and Appendix B. The source transcription in `original_explanation_verbatim.md` retains the original citation tokens.

Yes—the demonstrator implements a fresh, payload-bound authorization proof for each application request and each application response. That mechanism works in the supplied reference implementation. The code includes the complete payload in the authenticated statement, produces an actual Rosario–Wang masked witness, and checks both before permitting the operation. [D1] [D1]

What we have established is a working proof-per-message authorization model, supported by a conditional mathematical proof of credential privacy. That is narrower than proving that the complete HTTP6 replacement for HTTPS is already implemented and secure.

1. What I just verified

I reran the complete test suite: **all 69 tests passed**. I also ran another eight-message demonstration through the local HTTP carrier:

Operation or attempted attack	Freshly observed result
Initial synthetic human admission	Accepted
Root authorizes a distinct child	Accepted
Eight application requests	Eight fresh proofs accepted
Eight corresponding responses	Eight separate fresh proofs accepted
Replay of an accepted request	Rejected; no additional effect
Substitution of the destination	Rejected
Authenticated child requests prohibited export	Rejected
Response attempted under the wrong local channel	Rejected
Prepared request submitted after parent revocation	Rejected

That produced **18 accepted native proofs**: one admission, one minting operation, eight requests, and eight responses. It produced **exactly eight local application effects**.

These are actual reference-program results, not just expected behaviors written into the paper.

2. Yes: the proof covers the message payload

The implementation does **not** merely attach a login proof to an otherwise unprotected message.

Conceptually, each message contains:

$$\boxed{\text{Message proof object} = (x_e, Z_e, \tau_e)}$$

Here:

- x_e is the complete statement: sender, recipient, channel, direction, sequence, policy, operation, **actual payload**, nonce, and expiration.
- Z_e is the fresh Rosario–Wang masked membership witness.
- τ_e authenticates the statement and witness together.

The implementation calculates:

$$\tau_e = \mathrm{HMAC}_{a_v}(\mathrm{Encode}(\texttt{witness-envelope}, x_e, Z_e)).$$

The payload is inside x_e, so it is inside the authenticated bytes. The verifier also compares the submitted statement with the exact statement stored when that event was issued. **Changing the payload while keeping the old proof fails those checks.** [D3] [D1]

For example, suppose the authorized message says:

```
Sender:       worker-1
Operation:    read
Resource:     /reports/approved
Destination:  reports.example
Payload:      {"report": 7}
Sequence:     42
```

The existing proof cannot simply be moved to:

```
Payload:      {"report": 8}
```

or:

```
Operation:    export
Destination: outside.example
```

Those are different statements requiring their own valid authorization. Even a freshly authenticated request must still pass the inherited policy. The program checks that separately. [D1]

“Per message” means per protected application record—not automatically one proof per byte or per underlying network packet. A large streamed object would need a specified record/chunk format. The current attachment demonstrates complete bounded JSON records. [D4]

3. What the Rosario–Wang mathematics proves

The native response calculation is:

$$z_t = \left(\varphi(c_t(k_{r(t)})) + m_t\right) \bmod H.$$

The verifier knows the actual private masks and the enrolled reference. Subtracting the masks and reversing the private dictionary recovers the membership answers. Honest answers therefore pass by direct algebraic cancellation. Calibration and a separating schedule also permit authorized recovery of the enrolled string and dictionary. Those are mathematical results in the supplied formalization. [F1] [F1]

For the outside observer, the decisive result is:

$$\Pr[Z = z \mid X = x, G = g] = H^{-q}.$$

Every possible credential produces the same probability for each released word, because exactly one equally likely private mask vector explains that word for each candidate credential.

Therefore, observing the specified challenges and masked witnesses does not improve the observer's estimate of the original credential. It is not merely "too difficult to calculate." Under the stated masking and observation assumptions, the additional information is mathematically zero. [F1]

The Python distribution audit checks finite instances of that property exactly. The general result comes from the proof, rather than from extrapolating a successful test.

4. Does that continue after trillions of observations?

Yes, the supplied sequential privacy theorem covers any supported finite number of honest observations—including one trillion—when every event receives the required fresh private masks and preserves the specified observation boundary.

Its conclusion is:

$$\boxed{I(X; \mathscr{T}_1, \dots, \mathscr{T}_N \mid B) = 0.}$$

The proof multiplies credential-independent conditional likelihoods across the history. Consequently, the complete archive remains credential-independent. Increasing N does not, by itself, create leakage. [F1]

That statement protects the credential. It does not mean that every component of the message system has information-theoretic security.

Property	What supports it here
Privacy of the enrolled credential in the specified released transcript	Rosario–Wang's fresh-mask theorem
Integrity and authentication of the exact payload	The independent HMAC and context checks
Rejection of a second use	The atomic consumed-event ledger
Permission to perform the operation	The inherited-policy gate
Confidentiality of application content	A transport/encryption component still to be integrated

The attachment expressly makes those distinctions. HMAC is a keyed message-authentication mechanism; its security rests on cryptographic assumptions, rather than the fresh-mask secrecy theorem. [D3] [X1]

The fresh-entropy requirement is real

For the demonstrated 24-character RW-40 machine profile, each record consumes **150 fresh private mask bits**. Thus one trillion records require **18.75 TB of continuously packed masking material**, excluding other overhead. A request and response consume two allocations. A 65,536-bit tape supplies **436 complete records** under that profile. [D4]

The script stops when its tape runs out.

Your formalization explicitly proves that deterministic expansion of a fixed seed cannot create additional entropy. A short seed can support a separately analyzed **computational** implementation; it does not supply unlimited independent masks for this exact theorem. [F1]

5. Which HTTP6 claims are already demonstrated?

The demonstrated application mechanism is:

> An admitted child produces a fresh native witness for a particular message; the authenticated envelope binds that witness to the exact payload and context; the verifier checks current authority and consumes the event before allowing the protected effect.

The response performs the corresponding operation in the opposite direction and names the accepted request. The child uses its own reference, and the service uses its own reference. **The human does not have to solve another ceremony for every automatic message.** [D3]

This establishes an executable basis for your intended architecture. It also establishes that the mechanism need not search an expanding history to generate the next native proof: for the fixed tables, the native work is linear in the reference length. Full handling additionally processes the payload, ancestry, and durable state. [D3]

The earlier benchmark measured approximately **84 microseconds for native response generation with allocation** and **43 microseconds for native verification**. Those measurements excluded the network, HMAC envelope, database, hardware, encryption, and private provisioning. They demonstrate inexpensive local arithmetic—not completed Internet-scale throughput.

6. What remains to establish the full HTTP6 claim?

The present program is a **local designated-verifier model**. Its private provisioning places material into logical participant objects; its hardware identities are modeled attributes; its application payloads remain readable. It does not yet implement independently secured client/server processes, a native encrypted handshake, or a real trusted-screen ceremony. These are explicitly identified in the attachment, rather than silently counted as tested capabilities. [D4] [D4] [D4]

The next implementation milestone is therefore **two separately running endpoints**, with authenticated private-material provisioning, protected key use, encrypted application records, and durable state tested across failures. A hardware-bound profile must then exercise the real device attestation and trusted input/output path against live relay attempts.

The existing wrong-channel test demonstrates the **logical rejection rule** needed for a relayed-screen defense. It does not yet demonstrate that real hardware reliably supplies that rule's trusted local facts. The source itself treats live forwarding, active authentication, and complete observation handling as additional protocol obligations. [D3] [F1]

The conclusion

A proof per message payload: yes, implemented and exercised in the reference model.

No additional credential information from trillions of observations: mathematically established for the specified fresh-mask, complete-observation model.

An entirely information-theoretic, production-ready replacement for HTTPS: not established by this attachment. Its payload authentication is presently computational, its payload encryption remains to be integrated, and its deployment trust boundaries still need implementation evidence.

The strongest accurate statement is:

> **HTTP6's proposed per-message authorization mechanism is executable: each request and response carries a fresh Rosario–Wang witness bound to its exact payload, execution context, inherited authority, and one-use event. The supplied mathematics establishes passive credential privacy under fresh private masking; the demonstrator establishes the tested local authorization behavior.** [D3] [D3]

Appendix B. Reproduction and artifact index

B.1. Reproduction procedure

Extract the unchanged companion archive and enter its `HTTP6_RosarioWang_Demonstrator` directory. The script uses Python 3.10 or later and the standard library. Create new output directories for each demonstration rather than overwriting an earlier record. The following commands reproduce the relevant workflows; the benchmark command is optional and produces a new measurement rather than replacing the historical figure. [D4]

```
python3 -m unittest -v
python3 http6_rosario_wang_demo.py demo \
  --messages 8 --loopback --out review_loopback
python3 http6_rosario_wang_demo.py privacy \
  --out review_privacy.json
python3 http6_rosario_wang_demo.py capacity \
  --observations 1000000000000
python3 http6_rosario_wang_demo.py benchmark --records 10000
```

The reproduction for this edition used Python 3.13.5 on Linux x86_64. The suite completed 69 tests successfully. The loopback result recorded one synthetic admission, one child mint, eight requests, eight responses, 18 accepted native proofs, and eight local resource effects. The privacy audit returned exact equal-likelihood and posterior-preservation results for its enumerated finite cases. A trillion-event workload was not executed, and the benchmark in Section 5 remains the original recorded measurement. [D6; D7]

B.2. Artifact inventory

Package location	Purpose
`HTTP6_Per_Message_Proof_NIST_Review_White_Paper.pdf`	Typeset technical-review paper
`HTTP6_Per_Message_Proof_NIST_Review_White_Paper.md`	Editable source with LaTeX mathematics
`typesetting/`	Standalone LaTeX, build script, and publication styling
`original_explanation_verbatim.md`	Preserved explanation before audience editing
`support/HTTP6_RosarioWang_Python_Demonstrator.zip`	Unchanged executable review attachment, tests, schemas, and original evidence
`evidence/`	This edition’s regression log, loopback outputs, exact privacy audit, and reproduction receipt
`sources/`	Supplied formalizations, review appendix, source code, and mapping sources used by the paper
`manifest.json`	SHA-256 integrity inventory of packaged artifacts
`VALIDATION.txt`	Document, preservation, reproduction, and PDF checks

The unchanged demonstrator archive provides the complete test and source package. The separately copied `http6_rosario_wang_demo.py` is checked against the archive's corresponding script. Original source documents are preserved without editing. Generated observer files contain public records and synthetic application data, not enrolled private words, mask tapes, or authentication keys. The demonstration database is retained as local effect evidence, with its exact scope described in the README. [D4]

B.3. Claim-to-evidence concordance

Original explanation	Formal presentation	Supporting evidence
What was verified	Section 2	Regression log and local workflow report [D5, D6]
Proof covers the payload	Section 3	Complete statement and envelope code [D1, D3]
Mathematical correctness and privacy	Section 4	Native and sequential proofs [F1, F2]
Trillion-observation consequence and resource cost	Sections 4.3 and 5	Sequential theorem and capacity accounting [F1, D3, D4]
Demonstrated HTTP6 claims and primitive efficiency	Sections 5 and 6	Source, workflow, and historical benchmark [D1, D6, D7]
Full implementation milestone	Section 7	Explicit implementation boundaries and active-security model [D3, D4, F1]
Final scoped conclusion	Section 8	Combined source and evidence record

References and source register

The source register identifies supplied materials and their specific evidentiary roles. Mathematical claims use their stated hypotheses. Artifact test records describe the reference experiment. The existing NIST crosswalk supplies author-proposed relationships, not new normative requirements or program-specific submission instructions. No new external standard requirement is introduced by this editorial conversion.

[D0] Preceding explanatory response, *What the demonstrator proves and whether HTTP6 supports a proof per message payload.* Preserved in Appendix A and in `original_explanation_verbatim.md`. Editorial source, not an independent assessment.

[D1] `http6_rosario_wang_demo.py`. Reference demonstrator version `http6-rw-review-1.0`. In particular, `Peer.respond`, `Runtime.challenge`, `Runtime.submit`, `verify_native`, `recover`, `PrivateTape`, and `Ledger`. Supplied executable source, preserved unchanged.

[D2] `HTTP6_RosarioWang_Python_Demonstrator.zip`. Supplied runnable attachment containing the source, 69 regression tests, message schema, source map, exact finite audit, public records, and original measurement files. Archive retained unchanged in `support/`.

[D3] Rosario, Frank Dylan. *HTTP6: Executable Rosario–Wang Proof and Observer Demonstration.* Reference demonstration 1.0, November 9, 2026. Supplied `HTTP6_RosarioWang_Executable_Appendix.md`, particularly §§A–I and Equations A1–A10.

[D4] *HTTP6 / Rosario–Wang: executable review attachment.* Supplied `HTTP6_RosarioWang_Demo_README.md`; corresponding `README.md` in [D2]. Reproduction, profile definitions, function map, resource budget, observer model, and deployment boundaries.

[D5] `HTTP6_RosarioWang_Demo_Validation.txt`. Original attachment-validation record. Reports 55 artifact checks, 69 tests, finite distribution counts, local workflow, and a three-request original loopback run. This original three-request carrier record is distinct from the eight-request reproduction in [D6].

[D6] *Technical review edition reproduction record.* New execution of the unchanged [D2] source during preparation of this paper. Files: `evidence/unittest.txt`, `evidence/loopback/report.json`, `evidence/privacy.json`, and `evidence/reproduction_receipt.json`. Records 69 passing tests and the eight-request/eight-response loopback result. The receipt identifies versions and hashes. Local reproduction is not independent laboratory review.

[D7] `results/benchmark_10000.json` in [D2]. Original 10,000-record primitive-core benchmark: 83.7955048 microseconds for response generation with allocation; 43.1669605 microseconds for verification. Excludes network, TLS/AEAD, HMAC envelope, SQLite, hardware, and private provisioning. Preserved as `evidence/original_benchmark_10000.json`.

[F1] Rosario–Wang. *Consolidated construction, proofs, worked examples, nonce entropy, burn state, and proof-of-knowledge requirements.* Consolidated technical document, September 16, 2026. Supplied `Rosario_Wang_Consolidated_Formalization.md`. Principal locations: §§3–5 (native correctness, privacy, recovery), §7 (entropy), §8 (event burn), §9 (sequential and complete-view privacy), and §§11–14 (knowledge, active security, and review obligations).

[F2] *A Unified Geometry of Private Navigation and Observable Projection.* Version 1.0, September 13, 2026. Supplied `Unified-Geometric-Formalization.md`, particularly §§3–6. Establishes finite fiber/slice and conditional full-mask privacy relationships under explicit hypotheses.

[F3] *Appendix B · Lean 4 formalization registry.* Supplied `B-lean-formalization-registry(1).md`. Cross-references source declarations and distinguishes source correspondence, finite enumeration, and kernel validation of a specified checkout. No new Lean run is asserted by this paper.

[M1] Rosario, Frank Dylan. *HTTP6 and NIST: Human-Originated Identity, Hereditary Authority, and Security Control Enforcement.* Expanded Edition 2.0, November 9, 2026. Supplied `HTTP6_NIST_Requirements_and_Control_Mapping.md`, particularly §§2–5, 8, 12–14, and 21. Author-proposed architecture mapping and assessment-evidence framework.

[X1] Krawczyk, H., M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* RFC 2104, February 1997. External reference retained from the original explanation for the role of HMAC; not a validation of this composition. https://www.rfc-editor.org/info/rfc2104/

TECHNICAL WHITE PAPER

HTTP6

Human-Originated Authority for NIST-Aligned Cybersecurity

Hereditary authority, private-reference identity,
and a complete review-artifact specification

Frank Dylan Rosario

Expanded Public Review Edition 2.0

November 9, 2026

A human decision remains the origin of permission. Each descendant has its own identity, inherited limits, and verifiable provenance.

12 chapters / 12 embedded appendices

Models, schemas, protocol algorithms, reference source, examples, and review evidence

Contents

Review edition and reading guide **6**

Abstract **7**

1. Introduction: hereditary authority without disclosing the human commitment **8**

2. Contribution, method, and evidence basis **12**
2.1. Submission artifacts as an inspectable argument 12

3. Architecture: from the human pilot light to protected effects **14**
3.1. Human admission and geometric interpretation 14
3.2. One decision at every protected resource 14
3.3. Private origin, public reviewable provenance 15

4. Formal mechanisms and their security consequences **16**
4.1. Replay resistance through atomic consumption 16
4.2. Phishing resistance through intended-context binding 16
4.3. Least privilege through monotone delegation 17
4.4. Shared budgets across the process family 17
4.5. Immutable policy meaning and measured admission 17
4.6. Bounded cessation after the pilot light stops 18
4.7. Conditional privacy of the witness history 18
4.8. Authentication of minting and provenance 18

5. Policy, identity, and zero-trust integration **20**
5.1. Schema and decision pattern 20

6. Examples across all twenty SP 800-53 control families **22**
6.1. Technical enforcement and accountable operation 22
6.2. Lifecycle, response, and independent review 22
6.3. People, facilities, and organizational authority 23
6.4. Privacy, risk, acquisition, and suppliers 24
6.5. A common example-to-evidence pattern for every family 24

7. Assurance classifications and organizational outcomes **25**
7.1. Identity proofing, authentication, and federation 25
7.2. Cryptographic modules and private material 25
7.3. Security categorization and controlled information 25
7.4. CSF 2.0 and the lifecycle of responsibility 26
7.5. Lifecycle, privacy, and agent-risk programs 26
7.6. Separate assurance dimensions in the instance profile 27

8. Worked demonstration: a reporting-agent family **28**
8.1. Inspect the birth, action, and outcome as separate records 29

9. From classification coverage to a complete assessment record **30**
9.1. Define a finite requirement universe 30
9.2. Measure assignment and verification separately 30

9.3. Close a requirement through its objectives . . . 31
9.4. Detect omissions against an independent inventory . . . 31
9.5. Carry evidence through dependencies . . . 31
9.6. Reassess the requirements affected by change . . . 32
9.7. Exchange artifacts and an independently sourced inventory . . . 32

10. Bind the requirement profile to inherited authority 33
10.1. Preserve requirement identity without exposing commitment content . . . 33

11. Deployment and evaluation program 35
11.1. Review deliverables and reproducible execution . . . 35

12. Conclusion 37
12.1. An explicit public claim and an explicit review path . . . 37

Appendix A. Review model, trust boundaries, and implementation contracts 38
A.1. Objects and terminology . . . 38
A.2. Five security boundaries . . . 38
A.3. Artifact-possession and adversary matrix . . . 38
A.4. Protocol role contracts . . . 39
A.5. Protected source code submission inventory . . . 40
A.6. Functional and security acceptance . . . 40

Appendix B. Binary-possession privacy and hereditary authority propositions 41
B.1. The complete observer view . . . 41
B.2. Sequential independence theorem . . . 41
B.3. A sufficient exact private-label realization . . . 42
B.4. Provisioning a trillion-observation exact profile . . . 42
B.5. Exposed deterministic interpretation as a diagnostic case . . . 43
B.6. Secrecy and heredity compose through separate mechanisms . . . 43
B.7. Required proof and evidence deliverables . . . 44

Appendix C. Complete schemas and typed data structures 45
C.1. Authority_Policy_Schema.json . . . 45
C.2. protocol_contracts.schema.json . . . 58
C.3. review_objects.schema.json . . . 65

Appendix D. Protocol algorithms and state transitions 72
D.1. Enroll and prepare a human-originated root . . . 72
D.2. Ignite after the private-reference proof . . . 72
D.3. Mint a child directly or through an approved broker . . . 73
D.4. Verify a complete ancestry . . . 73
D.5. Authorize and commit the effect . . . 73
D.6. Canonical destination and information-flow decision . . . 74
D.7. Revoke, recover, and renew . . . 74
D.8. Assess a requirement and respond to change . . . 74

Appendix E. Executable reference implementations and complete test source 76
E.1. authority_model.py . . . 76
E.2. privacy_model.py . . . 80
E.3. state_model.py . . . 81

E.4. import_oscal_catalog.py 82
E.5. check_requirement_closure.py 83
E.6. run_public_example.py 86
E.7. test_authority_model.py 87
E.8. test_catalog_import.py 90
E.9. test_closure_checker.py 90
E.10. regenerate_public_example.py 91

Appendix F. Full policy examples and public signed-record fixtures 93
F.1. Authority_Policy_Example.yaml 93
F.2. review_policy.json 95
F.3. review_root_grant.json 95
F.4. review_child_grant.json 96
F.5. review_public_registry.json 96
F.6. review_request.json 97
F.7. review_action_proof.json 97
F.8. review_evidence_record.json 97
F.9. canonical_test_vector.json 98

Appendix G. Complete raw examples for the twenty control families 99
G.1. AC — Access Control 99
G.2. AT — Awareness and Training 100
G.3. AU — Audit and Accountability 101
G.4. CA — Assessment, Authorization, and Monitoring 101
G.5. CM — Configuration Management 102
G.6. CP — Contingency Planning 103
G.7. IA — Identification and Authentication 104
G.8. IR — Incident Response 105
G.9. MA — Maintenance 105
G.10. MP — Media Protection 106
G.11. PE — Physical and Environmental Protection 107
G.12. PL — Planning 108
G.13. PM — Program Management 109
G.14. PS — Personnel Security 109
G.15. PT — Personally Identifiable Information Processing and Transparency 110
G.16. RA — Risk Assessment 111
G.17. SA — System and Services Acquisition 112
G.18. SC — System and Communications Protection 112
G.19. SI — System and Information Integrity 113
G.20. SR — Supply Chain Risk Management 114

Appendix H. Complete CSF 2.0 outcome case catalog 116
H.1. GV.OC — Organizational Context 116
H.2. GV.RM — Risk Management Strategy 116
H.3. GV.RR — Roles, Responsibilities, and Authorities 117
H.4. GV.PO — Policy 117
H.5. GV.OV — Oversight 118
H.6. GV.SC — Cybersecurity Supply Chain Risk Management 118
H.7. ID.AM — Asset Management 119

H.8. ID.RA — Risk Assessment . . . 119
H.9. ID.IM — Improvement . . . 120
H.10. PR.AA — Identity Management, Authentication, and Access Control . . . 120
H.11. PR.AT — Awareness and Training . . . 121
H.12. PR.DS — Data Security . . . 121
H.13. PR.PS — Platform Security . . . 122
H.14. PR.IR — Technology Infrastructure Resilience . . . 122
H.15. DE.CM — Continuous Monitoring . . . 123
H.16. DE.AE — Adverse Event Analysis . . . 123
H.17. RS.MA — Incident Management . . . 124
H.18. RS.AN — Incident Analysis . . . 124
H.19. RS.CO — Incident Response Reporting and Communication . . . 124
H.20. RS.MI — Incident Mitigation . . . 125
H.21. RC.RP — Incident Recovery Plan Execution . . . 125
H.22. RC.CO — Incident Recovery Communication . . . 126

Appendix I. Assurance, categorization, CUI, and lifecycle cases 127
I.1. Security categorization and minimum security areas . . . 127
I.2. Identity assurance: raw cases for proofing, authentication and federation . . . 129
I.3. Cryptographic module assurance and private material . . . 132
I.4. CUI requirements: all seventeen Revision 3 families and selected enhancements . . . 136
03.01 — Access Control . . . 136
03.02 — Awareness and Training . . . 136
03.03 — Audit and Accountability . . . 137
03.04 — Configuration Management . . . 137
03.05 — Identification and Authentication . . . 137
03.06 — Incident Response . . . 137
03.07 — Maintenance . . . 138
03.08 — Media Protection . . . 138
03.09 — Personnel Security . . . 138
03.10 — Physical Protection . . . 139
03.11 — Risk Assessment . . . 139
03.12 — Security Assessment and Monitoring . . . 139
03.13 — System and Communications Protection . . . 139
03.14 — System and Information Integrity . . . 140
03.15 — Planning . . . 140
03.16 — System and Services Acquisition . . . 140
03.17 — Supply Chain Risk Management . . . 141
I.5. Lifecycle, organizational maturity, privacy and agent risk . . . 142

Appendix J. Assessment dossier, source inventory, and evidence exchange 146
J.1. Review-specific submission manifest . . . 146
J.2. Draft system implementation record . . . 147
J.3. Draft assessment plan and result linkage . . . 147
J.4. Corrective action and integration assignments . . . 148
J.5. OSCAL exchange map . . . 148
J.6. Independently obtained source inventory . . . 149
J.7. Cryptographic and private-reference evidence checklist . . . 149
J.8. Program-specific review gate . . . 149

Appendix K. Executed reference checks and production acceptance plan **150**
K.1. Actual bounded observations in this edition 150
K.2. Reproduce the local source checks 151
K.3. Production acceptance campaign 151
K.4. Additional construction and integration cases 152

Appendix L. Provenance, changes, and reproducible assembly **154**
L.1. Source basis and preservation 154
L.2. Review changes that preserve technical meaning 154
L.3. Reference verification and source catalog status 154
L.4. Build and reproducibility 155

References **156**
References added for the expanded review edition 158

Review edition and reading guide

Frank Dylan Rosario
Expanded public review edition 2.0 — November 9, 2026

This edition expands the public white paper into a self-contained technical review volume. The original fifteen main equations remain, the introduction is replaced with 2,000 words of paragraph-based exposition, and every main chapter has an accompanying implementation or review pattern. Appendices contain the schemas, algorithms, public record examples, source listings, raw classification examples, and bounded test observations used by the paper.

The Human Admission Client is the human-facing client component. The HTTP6 reference architecture comprises the underlying cryptographic, operating-system, and infrastructure components: Proof Authority, Enrollment Service, Minting and Delegation Service, Delegation Broker, Policy Decision Point, Server Circuit Twin, Authoritative State Plane, and Consumption Ledger. Rosario–Wang identifies the private-reference construction discussed in the source manuscripts. The main paper and original sources distinguish the specified mechanism from a completed implementation assessment.

Reading path. Chapters 1–4 develop the private human origin, hereditary delegation, and formal propositions. Chapters 5–8 connect policy and NIST classifications to concrete examples. Chapters 9–11 establish the requirement and evidence model. Chapter 12 states the public conclusions. Appendix B contains the binary-possession privacy theorem; C contains schemas; D contains production-integration pseudocode; E contains executable reference source; F contains generated public records; G–I contain the full examples; J–L organize review and provenance.

Evidence vocabulary. A mathematical proposition is established under its stated premises. An illustrative fixture specifies expected behavior and keeps production observations empty. An executed reference test records actual behavior of the bounded source model included here. A production assessment applies to its actual selected boundary, implementation, objectives, and evidence. These labels remain attached to their artifacts throughout the volume.

The review-specific additions are proposed specifications and original reference implementations. They are not presented as recovered proprietary engine source, deployed hardware protection, an official NIST mapping, or a certificate. The production construction and platform evidence are assigned to named review interfaces and owners. This structure enables a rigorous submission without copying the human commitment into descendants or inventing results for unexamined components.

Abstract

The HTTP6 reference architecture proposes hereditary authority: a human authorization event admits a protected root, and distinct child circuit instances inherit authenticated permissions bounded by their ancestry. The Human Admission Client supplies the human-facing experience; the Rosario–Wang construction supplies the private-reference admission interface; the Policy Decision Point, Minting and Delegation Service, and Authoritative State Plane supply policy-bound execution, minting, synchronized state, and resource enforcement. Standard delegation carries a receipt of authorized origin rather than the human's original commitment.

This expanded public paper develops that model for review against selected NIST guidance and requirements. It explicitly includes an acquired binary in the observer's knowledge. Under sequential conditional independence of complete observations, the resulting information-theoretic proposition establishes zero incremental information about the commitment for any supported finite history, including a trillion ceremonies. An independent-mask example, randomness accounting, and exposure diagnostics make the premises inspectable. Separate authenticated-delegation and policy-intersection arguments establish provenance and non-expanding authority.

The paper embeds schemas, protocol algorithms, state models, public signed-record examples, executable reference source, test results, and the full source-derived control-family and classification catalogs in its appendices. Chapters pair NIST-related failure scenarios with architecture mechanisms and acceptance conditions. Local reference tests establish the reported behavior of the bounded models; production private-reference implementation, hardware appraisal, deployed enforcement, and organizational evidence retain explicit review assignments. A version-pinned requirement inventory and evidence register connect each deployment obligation to an accountable implementation and assessment path.

Keywords: HTTP6; Human Admission Client; Rosario–Wang; hereditary authority; private-reference authentication; binary-possession privacy; process provenance; delegated agents; immutable policy; NIST; assessment evidence.

1. Introduction: hereditary authority without disclosing the human commitment

Digital identity becomes more consequential when software acts after the person who authorized it has stopped interacting. An administrator can initiate a service, that service can create workers, and those workers can invoke tools across several organizations. The original decision may be narrow: prepare a report, reconcile an account, or operate an approved service. The resulting execution can nevertheless span many machines and persist through many transactions. Hereditary authority makes that relationship explicit. Each executing instance receives a distinct identity and an authenticated grant whose permissible effects remain bounded by the decision from which it descends. Authority travels; the human's secret does not. [S1]

The HTTP6 reference architecture develops this principle through the human pilot light. Its Human Admission Client supplies the ceremony, the Rosario–Wang construction supplies a private-reference admission relation, and its Minting and Delegation Service preserves the resulting authorization in protected execution. A successful ceremony activates a particular root instance under a particular policy. That root may delegate approved work to children, and eligible children may obtain narrowly scoped authority to create further descendants. Each birth records the issuer, recipient, execution identity, policy, lifetime, and original authorization epoch. A verifier can consequently ask who authorized this process, which ancestors constrain it, and whether its current request remains permitted. [S1; S2]

The foundational design choice is to distinguish proof of an enrolled relationship from public disclosure of the information underlying that relationship. During the proposed standard ceremony, the person supplies membership witnesses interpreted through the authorized geometric construction rather than transmitting the remembered commitment as ordinary password text. The verifier evaluates the required relation and produces an admission decision. The decision authorizes bounded key use and process creation; it does not turn the human commitment into a reusable infrastructure credential. This separation makes private-reference admission a well-matched candidate for hereditary authorization, because descendants need an authenticated origin rather than copies of the originating secret. [S1, II.3]

The privacy proposition has a precise mathematical meaning. An observer starts with some knowledge, including any acquired binary circuit, its inspectable contents, and public information. The observer then records successful ceremonies. When each complete observable ceremony is conditionally independent of the human commitment, given that initial knowledge and the previous observations, the record supplies zero additional information about the commitment. The conclusion applies to any supported finite number of ceremonies. One trillion observations therefore contribute zero additional information under the same premises as one observation. This is a conditional information-theoretic result, not a numerical extrapolation from a short experiment. [S1, II.4]

Including the binary in the observer's initial knowledge is essential to the statement requested here. The review model names that acquired artifact explicitly and inventories every exposed byte. Program instructions, embedded geometric parameters, copied seeds, serialized enrollment material, and recoverable state all enter the observer model when acquisition exposes them. The proof then evaluates the complete subsequent transcript against that expanded view. Protected private interpretation or fresh secret randomness receives a separate custody description. The result is an inspectable artifact-possession proposition: observations leave the observer's posterior distribution equal to its prior distribution conditioned on the actual material already possessed. [D1]

The original source provides a sufficient ideal construction using fresh private labels. A hidden membership index is combined with an independent uniform mask, so every possible reported label has the same probability for every supported secret. Fresh masks across ceremonies preserve that equality across the history. An exact implementation of this profile supplies and accounts for the required private randomness outside the acquired observer view. The computational profile instead uses protected keys and pseudorandom scheduling under its declared assumptions. Both profiles preserve the architectural separation between the human reference and machine authority; their security arguments and provisioning evidence remain individually identifiable. [S1, II.4.2–II.4.3]

This formulation also explains what the introduction's trillion-observation statement asks reviewers to establish. The full view includes displays, messages, success indicators, timing categories, and any other admitted outputs. The same equality must hold after prior observations, rather than merely making each isolated response appear balanced. The appendix supplies a finite exact enumeration that illustrates the masking argument and contrasting exposed-interpretation cases that test its premises. It also records the randomness budget of the ideal profile. The implementation review can therefore examine the mechanism producing independence instead of treating a large binary, a rotating display, or a large observation count as the proof. [D1]

Hereditary authority uses this privacy foundation without confusing secrecy with permission. Zero additional information about a human commitment protects that commitment; authenticated delegation establishes which machine may act. A child proves possession of its own instance key, presents its parent-linked grant, and satisfies the inherited policy. Its public ancestry can establish provenance without containing the human's secret. The two properties reinforce each other: the origin remains private where required, while the origin of permission remains reviewable by authorized parties. The protocol separates commitment privacy, admission soundness, delegation authenticity, and resource enforcement so that each contributes a defined assurance. [S1, II.6; D1]

A circuit's ability to mint children is itself a delegated operation. A parent may hold a protected issuance service or may call an independently enforced minting broker. In either arrangement, the mint checks the parent's live ancestry, approved child template, requested policy, destination constraints, lifetime, and shared resource allowances before issuing a birth record. The new child receives its own identity and proves control of its corresponding key. A leaf circuit whose policy prohibits spawning remains a leaf. This structure supports extensive automation while keeping the right to manufacture additional authority inside the same permission system as every other consequential operation. [S1, III.10–III.11]

The policy rule is deliberately simple: a descendant's effective permission is the intersection of its own grant and every ancestor ceiling. An allowlist identifies eligible destinations and operations; a denylist preserves explicit prohibitions. A request must satisfy the combination, including object, tenant, data category, recipient, and contextual conditions. A worker authorized to publish an aggregate report does not thereby receive permission to export its source records. An approved service address does not authorize every operation available at that address. Because each generation adds constraints through conjunction, a valid chain can become more specialized without becoming more authoritative than its human-approved root. [S1, III.5–III.6]

Policy immutability extends to meaning rather than merely file contents. The admitted record commits to canonical policy bytes, role definitions, assignment snapshots, normalizers, action registries, and evaluator versions. An application may edit a local configuration, but independently enforced resources continue recognizing the authenticated admission. Planned change creates a new approved policy and instance generation. This makes ordinary maintenance compatible with stable authority: software can be patched, credentials can be renewed, and workers can be replaced while

the previous grant remains historically identifiable. The review asks both whether policy identity is protected and whether the executed action preserves the admitted interpretation. [S1, III.7–III.13]

Fresh proof accompanies each protected effect. The proof binds the acting instance, current epoch, intended peer, channel, policy, canonical request, and one-use position. A durable consumption record separates a retry from permission to create another outcome. Synchronized twins maintain authenticated logical state and recognize the next permissible transition. Their relationship is not an assumption that an attacker cannot match a timestamp; it is an agreement enforced through keys, context, sequence, and authoritative state. A recorder can copy a historical artifact without thereby obtaining the current, action-specific authority needed to substitute a request or produce an additional protected effect. [S1, II.5–II.6]

This is especially important for autonomous agents. A language model may propose an operation because of a user instruction, retrieved document, tool response, or maliciously injected text. Permission decisions occur outside that proposal-generating process. The agent submits a canonical request, and a protected monitor evaluates the request against the original human-approved scope. Persuasive content can influence a proposed plan while remaining unable to rewrite the authenticated policy or create a valid new delegation. Appropriate evaluation of the agent's accuracy and intended use remains a separate responsibility, alongside enforcement of which effects it is authorized to cause. [S2, §31.5]

NIST's guidance provides several complementary perspectives on this architecture. The access-control catalog asks whether permissions are authorized and enforced. Digital identity guidance asks how authenticator control, replay resistance, phishing resistance, and lifecycle management are established. Zero-trust guidance places decisions around resources and current evidence rather than network location alone. Assessment guidance asks what examination, interviews, and tests support the implementation statement. Identifiable lineage and a common resource gate connect these concerns. The contribution is not that every NIST objective is an authentication problem, but that many distinct objectives can share an accountable enforcement and evidence structure. [N1–N6]

The broader control families remain concrete operational responsibilities. Training eligibility can condition root admission after actual instruction. A personnel departure can retire the affected roots and descendants. A facility alarm can hold sensitive key release under the approved response plan. Supplier approval can govern which executable templates the mint accepts. Each integration authenticates a fact and assigns an accountable provider; the corresponding activity still has its own evidence. This organization lets reviewers distinguish native cryptographic mechanisms, integrated technical services, inherited safeguards, and organizational operations while tracing how all four influence the same deployed authorization boundary. [S2, §§24–25]

The NIST review package must therefore contain more than a persuasive narrative. It needs protocol roles, trust boundaries, lifecycle states, schemas, canonical encodings, algorithms, test cases, expected outcomes, evidence records, and responsible owners. This edition embeds those materials in its appendices and supplies editable files for independent inspection. Each chapter connects a recognizable failure scenario to a NIST objective, then identifies the response and an executable or operational acceptance condition. The source listings implement review models for selected mechanisms, while the private-reference engine and hardware appraisal remain explicitly named integration interfaces requiring their own implementation evidence. [D2]

The review package also protects the distinction between classification coverage and demonstrated satisfaction. A complete family index shows breadth. A version-pinned requirement inventory establishes the actual selected obligations. An implementation register assigns each obligation to mechanisms, parameters, providers, and assessment objectives. Evidence establishes what the de-

ployed system did under examination. Binding the approved requirement profile into the root and descendant records preserves that selection through automation, but a signed requirement identifier does not perform the required safeguard. The accompanying record checker detects missing assignments and inconsistent evidence structures; qualified reviewers determine whether the underlying artifacts support the claimed outcomes. [S2, §§23, 32]

The intended consequence is a change in what a captured object means. A copied binary is material whose disclosure must be included in the privacy analysis. A recorded login is historical evidence, not a standing instruction to authorize another effect. A child grant identifies a particular descendant, not a portable replacement for the human. A retained backup records earlier state, not a right to revive consumed positions. A stopped parent ends issuance and renewal according to an explicit fencing and lease policy. These distinctions turn the pilot-light metaphor into a system whose authority has an origin, a scope, a lifetime, and a verifiable continuation. [S1]

For reviewers, the central question becomes whether every claimed relationship survives implementation. Does the exposed artifact leave the required private interpretation outside the observer's view? Does the transcript distribution preserve the stated independence? Does the mint reject unauthorized births? Does the resource reject broader child policy, changed destinations, retired ancestors, and consumed positions? Do recovery and concurrency preserve the same rules? The following chapters and appendices make those questions reproducible through explicit premises, inspectable structures, and paired positive and adverse examples. The objective is a technically grounded account of hereditary authority whose evidence is as traceable as its delegations. [D1; D2]

For institutions, the practical opportunity is to connect human responsibility with machine scale without distributing the human commitment throughout the infrastructure. For developers, it is to build applications around narrow, verifiable requests instead of ambient credentials. For assessors, it is to follow one decision from enrollment through delegation to the protected outcome and supporting evidence. The Rosario–Wang construction supplies the proposed private origin; hereditary delegation supplies the disciplined continuation. Together, under the specified privacy and enforcement profiles, they provide a candidate foundation for automation that grows in capability through specialization while preserving the authority, accountability, and confidentiality of its original human decision throughout execution.

2. Contribution, method, and evidence basis

The technical basis is the unified architecture manuscript and its expanded NIST mapping. The manuscript defines geometric reconstruction, private-reference verification, protected circuit identities, authenticated delegation, synchronized execution, and immutable policy. The mapping adds control relationships, illustrative cases, four implementation modes, and an explicit process for completing requirement assignments and assessment evidence. This white paper reorganizes that material for technical, institutional, and public review. [S1; S2]

Three linked contributions organize the analysis. First, the architecture provides a common resource-level decision for humans, workloads, and delegated agents. Second, its formal rules make specific consequences inspectable: a consumed position cannot be consumed again, a child intersection cannot exceed the parent set, and an authenticated policy commitment identifies the exact admitted meaning. Third, the assurance model connects each selected requirement to its owner, implementation, parameters, dependencies, and evidence.

The paper uses four implementation modes. A **native mechanism** is defined in the architecture, such as one-use consumption or policy intersection. An **integrated technical service** supplies a separately specified capability, such as identity proofing or endpoint monitoring. An **inherited control service** supplies safeguards for a defined provider scope, such as hosting or a cryptographic module. An **organizational operation** supplies activities such as training, personnel decisions, procurement, and risk review. Several modes can contribute to one requirement. [S2, §24.2]

Mathematical statements below retain their stated premises. Illustrative cases specify expected behavior and the evidence that would establish an implementation result. The full source catalog, including unexecuted examples and their empty observation fields, accompanies this paper as appendices and accompanying source files. This preserves the distinction between an architectural argument, a test specification, and a completed assessment throughout the submission. [S2, §§23–24, 32]

2.1. Submission artifacts as an inspectable argument

Why the pattern matters. Assessment procedures examine an implementation and its evidence, not merely whether a control identifier appears beside an architectural claim. A reviewable answer connects the obligation, failure scenario, enforcing component, actual procedure, and observed result. This edition therefore places the supporting structures inside the paper's appendices rather than requiring the reviewer to infer them from a supplement title. [N2; R2]

The new code supplies a testable model of signed ancestry, complete-request predicates, context binding, transactional consumption, and shared-budget metering. It does not impersonate the original Rosario–Wang engine. The enrollment proof and hardware appraisal remain explicit implementation interfaces. The same distinction applies to structural schema validation, a finite mathematical model, and evidence that a production system enforces the specified boundary. Appendix K records the results that were actually generated for the review models.

```
review_item:
  claim: child-authority-stays-within-root-ceiling
  model: ancestor-policy-conjunction
  implementation: reference_code/authority_model.py
  test: test_child_local_expansion_cannot_override_root
  result_scope: synthetic-reference-model
  production_result: not_assessed
```

```
REVIEW(claim):
  identify its source, premises, and precise observer/system boundary
  match the claim to a model, interface, implementation, and test
```

```
record the result at the level actually examined
retain unresolved integration obligations as named review items
```

The result is a source-to-evidence chain that can be challenged or reproduced without granting a stronger status than the artifact supports. Proposed material added in this edition carries references [D1] and [D2]; the original manuscripts remain separately preserved.

3. Architecture: from the human pilot light to protected effects

3.1. Human admission and geometric interpretation

The human-facing admission ceremony uses time-indexed geometry and membership responses interpreted through a protected enrollment relationship. Authorized components reconstruct the same geometry from the specified private material and challenge context. The Rosario–Wang verification interface evaluates the required relation and returns the admission decision. The Proof Authority binds that result to the intended verifier, instance key, root policy, authorization epoch, and one-use challenge. [S1, II.2–II.5; III.3–III.4]

The initial event is the **human pilot light**: a deliberate activation of bounded machine authority. Its acceptance authorizes protected key use and root admission. Descendants receive machine identities and scoped grants; they receive neither the human's remembered commitment nor a reusable representation of the human's authority. A current workload authenticator demonstrates machine participation under a delegation. The human authentication time remains a separate fact.

A circuit in this model is an executing authorization instance with measured software, protected identity material, a policy commitment, and current state. A parent and child may run the same software while retaining different keys and birth records. A twin is the designated peer that maintains corresponding authenticated channel state. These are logical and cryptographic relationships, made concrete through registration, attestation, sequencing, and resource checks. [S1, II.6; III.2, III.10–III.14]

3.2. One decision at every protected resource

$$\begin{aligned}\mathsf{Permit}(v,q) = \ & \mathsf{ChainValid}(v) \\ & \wedge \mathsf{InstanceValid}(v) \\ & \wedge \mathsf{Live}(v) \\ & \wedge \mathsf{PolicyAllows}(v,q) \\ & \wedge \mathsf{FreshProof}(v,q) \\ & \wedge \mathsf{ReserveAndConsume}(v,q).\end{aligned} \tag{1}$$

The instance v requests a normalized effect q. The first predicates establish its valid ancestry, admitted execution identity, and current authority. The next predicates establish permission for the complete request and a fresh action-bound authenticator. The final predicate reserves shared allowances and consumes the relevant one-use position. The conjunction operator, $\wedge$, requires every condition to hold. The formula restates the architecture's common enforcement rule. [S1, II.6.6; III.15; S2, M1]

The design consequence is complete mediation at the point of effect: database execution, key use, network transmission, process creation, or an agent tool call. AC-3 and AC-25 provide relevant access-enforcement and reference-monitor objectives; IA-9 addresses service authentication; SC-23 addresses session authenticity. Local containment and independent resource enforcement preserve the same decision across alternate access paths. A direct database request therefore confronts the authorization rule even when it bypasses an application frontend. [N1]

3.3. Private origin, public reviewable provenance

The original commitment remains an enrollment-side secret rather than a field in the child grant. A human-admission receipt contains the authorized root key, policy, epoch, intended context, and receipt identifier. The receipt can establish that the recognized admission authority approved that root without publishing the secret or a simple secret-derived password verifier. Child identities are independently generated, so distributing child certificates does not require distributing the human reference. [S1, III.3–III.4]

For the requested binary-possession statement, this edition strengthens the observer definition explicitly:

$$V_0 = (B, P_0), \qquad \left[O_\tau \perp S \mid (V_0, O_{<\tau})\ \forall\, \tau \le n\right] \Rightarrow I(S; O_{1:n} \mid V_0) = 0. \tag{H1}$$

B contains all bytes exposed by the selected circuit-acquisition experiment, P_0 the remaining initial information, S the human commitment, and O_τ the complete next observation. The conclusion follows by the conditional mutual-information chain rule. It applies to a trillion observations when the stated condition holds for the entire supported sequence. Appendix B proves the proposition, separates baseline binary disclosure from incremental disclosure, and supplies the independent-mask example.

The significance is a clear design contract: the public artifact and observation history are analyzed together, while any separately protected interpretation is inventoried rather than assumed. The source establishes this result conditionally and gives an ideal sufficient model. A production binary receives the corresponding exposure and composition analysis before the unconditional property is attributed to that implementation. The human-to-child provenance result separately follows from authenticated receipts and delegation checks. [S1, II.4; D1]

Why NIST asks for these distinctions. Authentication, key protection, and least privilege address different failures. A private transcript protects a secret; it does not by itself authorize a new process. A valid grant authenticates authority; it does not itself prove zero transcript leakage. The HTTP6 reference architecture assigns the admission engine, protected mint, and resource monitor the respective obligations. [N1, IA-5, AC-3, SC-12; N3]

```
observer_profile:
  artifact: complete-acquired-binary-and-extracted-fields
  observations: [challenge, display, witness, result, admitted-timing]
  privacy_claim: zero-incremental-information-under-H1
  baseline_binary_analysis: separately_required
child_grant:
  root_receipt_ref: opaque-approved-admission-receipt
  human_secret_field: absent
  parent_record_digest: bound
  child_key_and_policy: bound
```

```
OPEN_ROOT_AND_DELEGATE:
  accept only a context-bound human proof under the selected profile
  record an authenticated root admission without exporting the commitment
  mint each eligible child against the live parent and inherited policy
  require the child to prove its own current key possession
```

4. Formal mechanisms and their security consequences

4.1. Replay resistance through atomic consumption

$$\begin{aligned}\mathsf{Admit}(q,n,e) \iff\ & \mathsf{Verify}(\pi,c)\\ & \wedge\ \mathsf{CAS}(\mathsf{Ledger}[e,n],\mathsf{unused},\mathsf{consumed}),\\ & \#\mathsf{SuccessfulConsumptions}(e,n)\le 1.\end{aligned} \tag{2}$$

The epoch e identifies an authorization generation, nonce n identifies a challenge, and context c identifies its intended use. The proof π must verify before the ledger performs an atomic compare-and-swap, CAS. Once the state changes from unused to consumed, another request cannot perform the same transition. The final inequality follows from a consistent authoritative ledger. A durable operation record and idempotent execution connect that decision to the business outcome. [S1, II.5.2; S2, M2]

This supplies a specific response to the replay-resistance objective in SP 800-63B-4 §3.2.7 and IA-2(8). Capturing an accepted proof does not authorize a second consumption. A retry can retrieve the previously recorded result without creating another effect. The assessment case submits the same proof through concurrent replicas, interrupts the transaction, and retries after recovery. The required observation is one accepted consumption and one controlled outcome, with the actual resource inspected independently. [N1; N3]

4.2. Phishing resistance through intended-context binding

$$\begin{aligned}c &= \mathsf{Enc}\left(\begin{array}{c}\mathsf{verifier},\mathsf{origin},\mathsf{channel},h(q),\mathsf{id}_v,\\ h(P),e,n,T\end{array}\right),\\ \pi &= \mathsf{Auth}_{k_v}(\mathsf{Enc}(c,h(G),\mathsf{lineage})).\end{aligned} \tag{3}$$

The encoding binds the verifier, authenticated origin and channel, exact action digest, instance, policy, epoch, nonce, and expiration T. The authenticator also covers the reconstructed geometry G and lineage. The hash function h commits to canonical bytes; k_v identifies the protected authentication key. Under correct context construction, authenticator security, collision resistance, and unambiguous encoding, changing the intended destination or operation changes the authenticated statement. [S1, II.5.1–II.5.4; S2, M3]

SP 800-63B-4 §3.2.5 identifies verifier-name and channel binding as phishing-resistance mechanisms. The Proof Authority and the Server Circuit Twin apply those bindings to the human ceremony and the resulting machine actions. Trusted confirmation presents the consequential fields of the authenticated operation. A relay may carry genuine traffic while gaining no independent right to change its content or recipient. Tests therefore target cross-origin reuse, connection splitting, body substitution, and deceptive confirmation. Protected transport separately supplies payload confidentiality. [N3]

4.3. Least privilege through monotone delegation

$$\mathscr{A}_v = \mathscr{A}_u \cap \mathscr{A}(P_v) \quad \Longrightarrow \quad \mathscr{A}_v \subseteq \mathscr{A}_u \subseteq \mathscr{A}_0. \tag{4}$$

The parent is u, the child is v, and $\mathscr{A}(P_v)$ is the child's declared permitted-request set. Intersection, $\cap$, requires each child request to satisfy the parent restrictions and the child's own restrictions. The root begins inside its approved ceiling $\mathscr{A}_0$. Applying the subset relation at each generation proves by induction that every valid descendant remains inside that ceiling. The predicates apply to complete requests, preserving the relationship among operation, resource, destination, and context. [S1, III.5–III.6, III.17; S2, M4]

This is a direct architectural response to AC-3 and AC-6. A child cannot enlarge its grant by editing its local allowlist, removing a denial, or requesting a broader role from itself. Protected issuance checks the complete chain before admitting a new child. The operational significance for agents is clear: generated text proposes an action; an independent monitor determines whether the inherited permission allows the effect. The test suite attempts ancestor removal, unauthorized grandchildren, and privilege expansion through a broadly privileged intermediary. [N1]

4.4. Shared budgets across the process family

$$U_u + \sum_{v \in \mathsf{Children}(u)} R_{u,v} \le B_u. \tag{5}$$

B_u is the parent's total authorized allowance. U_u is its direct consumption, and $R_{u,v}$ is the amount committed to each child's subtree, including allocations already spent. Atomic reservations keep the sum within the original ceiling. Only safely fenced, unspent allocations are returned. A child's restart retains the authoritative accounting rather than creating a new copy of the allowance. The quantity can represent transactions, spending, exported bytes, concurrency, or admitted descendants. [S1, III.16; S2, M5]

The rule complements permission intersection with quantitative containment. A worker allowed to publish reports still operates within a shared family allowance. SC-6 provides a relevant resource-allocation relationship, while AC-6 connects to bounded use. Tests race sibling requests for the last allocation, interrupt reservations, and attempt resets through new instance identities. Capacity planning and denial-of-service safeguards remain part of the supporting platform; the equation establishes how delegated consumption stays inside its approved budget. [N1]

4.5. Immutable policy meaning and measured admission

$$\begin{aligned} h_{P_v} &= H(\mathsf{Enc}(\texttt{policy-v1}, \mathsf{Canonical}(P_v))), \\ M_v &= H\Big(\mathsf{Enc}(h_{\mathsf{code}}, h_{P_v}, h_{\mathsf{config}}, h_{\mathsf{launcher}})\Big). \end{aligned} \tag{6}$$

The first digest commits to the policy and its pinned interpretation. The second commits to the code, policy, configuration, and launcher admitted for the instance. Policy meaning includes the evaluator, role catalog, assignment snapshot, normalizer, and action registry. Attestation and protected registration connect those commitments to the executing instance. A controlled software update creates a newly measured generation with its own admission and retained provenance. [S1, III.7, III.9, III.13; S2, M6]

This connects CM change-control and configuration objectives with SI integrity and input-validation objectives. Changing what a role name means cannot silently expand an already admitted worker. Changing an adapter cannot silently reinterpret a previously denied request. A local file remains editable as data, while the independent monitor recognizes the authenticated commitment. Assessment compares actual measurements, tests altered role catalogs and modules, and verifies that the authorized operation matches the effect executed by the resource. [N1, CM-2, CM-3, CM-5, CM-6, SI-7, SI-10]

4.6. Bounded cessation after the pilot light stops

$$\Delta_{\mathsf{stop}} \le d_{\mathsf{fence}} + \ell_{\max} + \epsilon_{\mathsf{clock}}. \tag{7}$$

The stopping interval is bounded by the time d_{fence} for lease issuers to stop renewal, the maximum remaining lease $\ell_{\max}$, and accepted clock error $\epsilon_{\mathsf{clock}}$. The deduction assumes that renewals require current ancestry and no offline route can issue more leases after fencing. Online gates can reject earlier. Logical sequence and authenticated transitions maintain twin continuity; the design measures cessation rather than relying on infinitesimal timing differences. [S1, II.6.1–II.6.2; III.18; S2, M7]

The relationship makes AC-12 termination, SC-23 authenticity, and SC-24 safe-failure behavior testable. Root failure, subtree revocation, and stale state have defined consequences. Human-session timeouts remain independently enforced: machine activity demonstrates machine continuity and does not update the person's authentication time. The assessment stops a parent, delays propagation, applies supported clock skew, and records the last accepted descendant action against the admitted bound. [N1; N3, §5]

4.7. Conditional privacy of the witness history

$$\Big[\forall\, \tau :\ W_\tau \perp S \mid (P, W_{<\tau})\Big] \Longrightarrow I(S; W_{1:n} \mid P) = 0. \tag{8}$$

S is the enrolled secret, P the defined public view, and $W_{1:n}$ the observed witness history. Conditional independence at each ceremony makes each contribution in the mutual-information chain rule zero. The source gives an ideal fresh-private-label construction as a sufficient model. The public view includes relevant visible geometry and metadata; repeated-history conditioning makes the statement apply to the specified sequence rather than only a single response. [S1, II.4; S2, M8]

The proposition supports a precise authentication-secret exposure argument. The exact-privacy profile uses genuinely fresh private randomness with consumption accounting; the computational profile uses protected keying material and pseudorandom scheduling. Captured keystrokes, screens, timing, and logs receive explicit treatment in the chosen observer model. Broader personal-data processing receives purpose, retention, and access controls. These are complementary assurance questions: secrecy of the enrolled reference and appropriate handling of identifiable activity each retain their own evidence. [N3, §7; N10–N12]

4.8. Authentication of minting and provenance

An instance contains minting capability only when its admitted policy authorizes it, or accesses a protected minting service acting under that same scope. The mint authenticates the parent's request and names the child it is authorizing. Direct possession of code containing a mint routine is distinct from authority to use the protected issuer key. The proposed grant binds the ancestry to the original receipt:

$$D_{u\to v} = \mathsf{Sign}_{sk_u}\left(\mathsf{Enc}\left(\begin{array}{c} H(D_u), \mathsf{id}_v, pk_v, M_v, h(P_v), \\ h_{\mathscr{R}}, \mathsf{root}, e, T_v, \mathsf{birth}_v \end{array}\right)\right). \tag{H2}$$

The parent key sk_u authenticates a record containing its preceding grant digest, the child's identity and public key, measured execution, child policy, root requirement profile, root identifier, epoch, expiration, and birth identifier. The child proves possession of the bound private key. Complete verification checks every edge and the separately trusted root admission; a string naming a parent is not accepted as provenance.

This addresses service identity and least-privilege objectives together. A valid child can perform its permitted work without knowing the human commitment. A forged child, a transplanted certificate, a changed policy, or an expired parent fails an identified predicate. Appendix E supplies signed-chain reference code and negative tests, while Appendix D specifies the additional production appraisal, quota, and atomic birth steps. [N1, IA-4, IA-9, AC-3, AC-6; D2]

```
mint_request:
  parent: scheduler-instance
  child_template: approved-report-worker
  child_public_key: independently-generated
  preserve: [root-receipt, root-epoch, requirement-profile]
  constrain: [policy, lifetime, depth, destinations, operations, budget]
  prohibit: [human-secret-export, unrestricted-signing, caller-self-approval]
```

```
MINT(parent, child):
  verify parent ancestry and explicit mint permission
  appraise the child key and measured template
  reserve birth and resource quotas; recheck revocation at commit
  sign the bound grant and activate only the registered child
```

5. Policy, identity, and zero-trust integration

The policy engine evaluates a canonical request containing the subject instance, operation, target resource, destination, arguments, data classification, and current trusted context. Default denial and deny-overrides semantics preserve prohibitions through the ancestry. Missing required attributes and evaluation errors close the decision. An allowlist entry for a host identifies an eligible endpoint, while operation rules determine what may actually be done there. [S1, III.5–III.9; S2, §§9–10]

IAM and RBAC assignments are resolved into authenticated snapshots. Attribute-based conditions cover subject, object, action, and environment. Trusted providers supply authoritative facts such as personnel status, data classification, or admitted machine identity. The requester's assertion is a proposed input until that source validates it. Role definitions, context providers, and operation adapters are part of the admitted meaning. This connects role and attribute enforcement to AC-3(7), AC-3(13), AC-16, and SP 800-162. [N1; N9]

Destination control follows the actual connection. The broker checks normalized hostname, resolved address, service identity, protocol, port, and route requirements. Redirects receive another authorization decision. Payload rules distinguish an approved aggregate from raw customer records, including when both would use the same destination. An intermediary preserves the original caller's scope; its broader infrastructure credential remains protected inside the broker rather than becoming a substitute for the child's authority.

The mapping to SP 800-207 is functional. The policy engine evaluates current evidence and inherited ceilings. The policy administrator admits instances, establishes sessions, and retires generations. Enforcement points guard resources and brokered effects. Immutable policy fixes the maximum grant; dynamic posture, time, and revocation narrow its current exercise. This supports cloud and multi-location deployment without treating network placement as sufficient identity. [N6; N7]

Implementation can begin with a bounded workflow. The owner inventories all routes to its protected resources, installs the necessary adapters, and admits a limited set of workers. The success criterion is enforcement coverage of that boundary. Extending the architecture to additional services then expands both the resource inventory and the evidence obligations through an approved transition. [S1, III.13; S2, §11]

5.1. Schema and decision pattern

Why NIST asks for it. A permission expressed as a role name can change meaning when the role catalog changes; an approved host can expose both safe and destructive APIs. Access enforcement therefore needs the requested operation and object as well as a recognized identity. The Policy Decision Point pins the interpretation and evaluates the full tuple, including the current authoritative facts used by each condition. [N1, AC-3, AC-6, AC-16; N9]

The original full `http6.policy.v1` JSON Schema and its example are embedded in Appendix C. This edition also provides http6.review-policy.v1, a deliberately narrower schema used by the runnable reference model. The review schema supports exact allow matches, explicit wildcard denials, role-snapshot identity, child-template selection, mint permission, depth, and a metered budget. It is a separate named profile rather than a silent replacement of the richer original schema.

```
policy_pattern:
  default: deny
  conflict_resolution: deny_overrides
  ancestry: conjunction
  allowed_tuple: [report.publish, reports.example, daily, aggregate]
  blocked_tuple: [ledger.export, '*', '*', '*']
  role_catalog: pinned-at-admission
```

```
  unknown_attribute: deny
  changed_interpretation: require-new-admission
```

```
POLICY_ALLOWS(ancestry, actual_request):
  validate the typed, normalized request and trusted context
  for each admitted ancestor policy:
    require a matching full-request allow and every mandatory condition
    reject any matching deny or unresolved required attribute
  return PERMIT only for the unchanged request delivered to the resource
```

A policy schema validates representation. Semantic checks resolve referenced templates, actual service identities, expected roles, and parameter ranges. Cryptographic checks authenticate the selected record. Runtime checks enforce the result. This four-stage separation prevents a syntactically valid document from becoming its own authority.

6. Examples across all twenty SP 800-53 control families

The following family examples preserve the source mapping's breadth while presenting it as an implementation narrative. Identifiers such as `EX-AC-01` locate complete raw fixtures in Appendix G. The control anchors identify relevant objectives within a family. Every selected clause and enhancement is subsequently handled by the requirement register in Section 9. All examples describe expected behavior and the evidence required to establish it. [N1; N2; N16; S2, §25]

6.1. Technical enforcement and accountable operation

AC — Access Control (`EX-AC-01`). A reporting worker receives an instruction to export the customer ledger to an unrelated endpoint. Its ancestral ceiling permits only an approved aggregate. The broker and resource gate deny the export while permitting the authorized report. Reviewers examine role approvals, complete-request decisions, and the destination's actual effects. The case links AC-3, AC-4, and AC-6 to the shared enforcement rule.

IA — Identification and Authentication (`EX-IA-01`). A recorded ceremony is submitted with a consumed nonce or a different audience. The verifier rejects it; a fresh, correctly bound ceremony activates only the named root. Child proofs identify workloads, and machine heartbeats leave human authentication time unchanged. Evidence includes enrollment, authenticator configuration, channel-binding tests, replay tests, and recovery records.

SC — System and Communications Protection (`EX-SC-01`). An approved report connection redirects to an external peer. The broker checks the actual peer and requires new authorization before releasing sensitive content. Protected key services and runtime isolation support the channel. Evidence covers peer authentication, payload protection, broker-bypass attempts, capacity, and the selected cryptographic configuration.

CM — Configuration Management (`EX-CM-01`). A worker replaces its role catalog with a broader version and attempts to reuse the old admission. The independent gate continues recognizing the admitted digest. A legitimate update receives a reviewed replacement generation. Baseline manifests, change approvals, measurements, rollout records, and rollback tests establish the controlled transition.

SI — System and Information Integrity (`EX-SI-01`). An ambiguous encoded path exposes an adapter defect. Typed validation rejects the malformed request; the vulnerability program analyzes the defect and admits a corrected adapter with regression evidence. The deployment tracks the affected version and its retirement. Integrity, malicious-code protection, monitoring, and flaw remediation remain distinct contributions within the family.

AU — Audit and Accountability (`EX-AU-01`). A worker attempts to erase the record of its denied export. Its write-only submission role cannot edit retained evidence. Authorized reviewers trace the human admission, root epoch, worker, policy, requested effect, and outcome. Collector-failure exercises, protected storage, time quality, retention, and review records establish the audit service's operation.

6.2. Lifecycle, response, and independent review

CA — Assessment, Authorization, and Monitoring (`EX-CA-01`). A direct database account bypasses the documented broker. An independent assessment records the finding against the exact deployment rather than accepting a diagram as implementation evidence. Examination, interviews, and tests support an explicit authorization decision. The requirement remains visible until the responsible owner resolves the affected objectives.

CP — Contingency Planning (`EX-CP-01`). A root fails and an operator restores an older image. Recovery preserves consumed positions and business outcomes, then establishes a fresh approved generation. It resumes the authorized task without resetting historical authority. Evidence includes the business-impact analysis, backup verification, restoration exercises, recovery staffing, and measured recovery objectives.

IR — Incident Response (`EX-IR-01`). A compromised scheduler has issued several workers. The responder identifies its subtree, fences new issuance and renewal, preserves evidence, and checks cessation at the resources. Unaffected independently authorized work continues when policy permits. The response plan supplies triage, escalation, reporting, eradication, and recovery decisions alongside the technical containment.

MA — Maintenance (`EX-MA-01`). A technician receives a ticket-bound circuit permitting health inspection and an approved patch, while blocking data export and child creation. The circuit expires at the service-window boundary and the host receives post-service integrity review. Tool validation, technician authorization, session termination, and local servicing procedures complete the maintenance evidence.

MP — Media Protection (`EX-MP-01`). A retired drive remains in custody until its approved, device-specific sanitization method is performed and verified. The Policy Decision Point authorizes export and disposition operations and records the responsible identities. Actual media handling, verification, and release evidence establish completion under the selected sanitization guidance. [E7]

6.3. People, facilities, and organizational authority

AT — Awareness and Training (`EX-AT-01`). A cryptographically authenticated administrator lacks current root-operations training. The admission policy holds the privileged role until a trusted learning record establishes eligibility. The training owner supplies the curriculum, instruction, accessibility, assessment, and refresher evidence. The integration makes a verified people-program decision enforceable at admission.

PE — Physical and Environmental Protection (`EX-PE-01`). An authenticated facility controller reports a tamper alarm affecting the root host. The approved response suspends new sensitive key release and opens a review. The facility separately supplies locks, visitor procedures, power, fire protection, and environmental safeguards. Inspections and exercises establish those safeguards, with life-safety procedures retained by the appropriate systems.

PL — Planning (`EX-PL-01`). Architecture review discovers an unmediated backup-export route. The system plan adds the route, its data flow, owner, and required enforcement adapter. The unresolved path stays in the selected boundary and assessment inventory. Planning evidence therefore connects the written architecture to all usable paths rather than only the intended frontend.

PM — Program Management (`EX-PM-01`). An administrator proposes expanding an enterprise root beyond its organizational mandate. Executive governance establishes the accountable authority, funding, risk decision, and approved scope. A technical ability to ignite a root is evaluated against that organizational approval. Program plans, appointment records, review decisions, and metrics supply the supporting evidence.

PS — Personnel Security (`EX-PS-01`). A verified employee departure is linked to the affected roots and descendants. New issuance and renewal stop, role assignments are retired, and access removal is reviewed. A permitted transfer receives a newly approved role snapshot. Screening, agreements, personnel coordination, and revocation observations remain separately identifiable evidence.

6.4. Privacy, risk, acquisition, and suppliers

PT — Personally Identifiable Information Processing and Transparency (`EX-PT-01`). An analytics service requests contact information collected for account recovery for a different purpose. A valid service identity still encounters the approved-purpose restriction. The privacy program supplies the basis for processing, notices, retention, and relevant individual-rights procedures. The Policy Decision Point enforces the authorized use and retains minimized decision evidence.

RA — Risk Assessment (`EX-RA-01`). A discovered direct database path bypasses the lineage gate. The owner evaluates the consequential exposure, assigns treatment, removes or mediates the path, and retests it. Threat analysis also examines reference exposure, verifier compromise, concurrency, denial of service, and adapter defects. Risk closure follows the treatment evidence for the actual scenario.

SA — System and Services Acquisition (`EX-SA-01`). A supplier provides a signed policy-engine release but omits required testing and update evidence. Admission remains on hold until the acquisition deliverables are accepted. Contracts and development plans identify security requirements, logging behavior, provenance, tests, and resilient updates. The release signature supplies origin and integrity; the acceptance process supplies the assurance decision.

SR — Supply Chain Risk Management (`EX-SR-01`). An unapproved supplier component arrives with a valid signature. Supplier authorization and provenance review still determine eligibility. The organization evaluates criticality, contracting, inspection, notification, and offboarding duties. The Minting and Delegation Service binds admitted artifacts and service routes to the approved supplier relationships, and the Authoritative State Plane fences those relationships when retired.

6.5. A common example-to-evidence pattern for every family

Why the breadth matters. Technical policy enforcement, physical safeguards, training, supplier oversight, and risk authorization address different failure scenarios. The twenty family cases therefore name both the architecture's digital response and the owner responsible for the underlying safeguard. Their full raw input, expected positive and adverse behavior, and evidence fields are now embedded in Appendix G. An authenticated facility alarm establishes its origin; inspection and exercise establish that the facility safeguard exists and works. [N1; N2]

```
family_case_pattern:
  source: selected-NIST-control-and-exact-clause
  failure_scenario: observable-event-or-missing-safeguard
  implementation_mode: native-integrated-inherited-or-organizational
  owner: named-accountable-role
  positive_case: authorized-business-function
  adverse_case: specific-prohibited-or-invalid-transition
  evidence: actual-artifacts-for-each-applicable-objective
  observed_result: null
```

```
EVALUATE_FAMILY_CASE(case):
  identify the actual selected clauses behind the family label
  examine the responsible technical service or organizational procedure
  exercise the positive and adverse paths applicable to those clauses
  verify the resource effect or underlying operational evidence
  record each determination without promoting a family example to blanket coverage
```

The classification catalog supplies a review path for every family, and the requirement inventory supplies the finer denominator. The distinction lets an assessor identify an actual omission while still reusing the common admission, delegation, monitoring, and evidence patterns across multiple control areas.

7. Assurance classifications and organizational outcomes

7.1. Identity proofing, authentication, and federation

The SP 800-63 suite separates three assurance questions. Identity assurance concerns proofing and enrollment. Authentication assurance concerns the transaction that demonstrates authenticator control. Federation assurance concerns the relationship among the subscriber, issuer, relying party, and assertion. The Minting and Delegation Service preserves these distinctions when a human approval becomes a workload delegation. [N3–N5]

For **IAL1, IAL2, and IAL3**, the selected proofing provider completes the applicable evidence, validation, verification, enrollment, and operational requirements before the receipt is bound to the account. The source includes a separate pseudonymous case in which no proofing level is claimed. For **AAL1**, admission follows the selected permitted authenticator; **AAL2** integrates two distinct factors and the required phishing-resistant option; **AAL3** uses the required public-key profile, non-exportable key, phishing resistance, and other applicable requirements. [S2, §28]

For **FAL1, FAL2, and FAL3**, the relying party evaluates the selected trust, assertion, audience, transaction-binding, and authenticator requirements. A child signing key identifies that workload; the subscriber's federation relationship is established through its own required procedure. The fixtures include wrong audiences, injected assertions, and human-session renewal attempted with only a machine heartbeat. Notifications, recovery, usability, accessibility, and redress remain part of the selected profile. [N3–N5; S2, §28]

7.2. Cryptographic modules and private material

The cryptographic profile records assessed entropy, random-bit generation, purpose-separated keys, cryptoperiods, custody, retirement, and compromise response. Geometry provides reproducible structure; private randomness and selected cryptographic services provide the security properties assigned to them. The exact-mask profile accounts for fresh private randomness independently of the computational scheduling profile. [N10–N12; S1, II.2–II.4]

FIPS 140-3 module assurance attaches to the actual component, its certificate, security policy, version, operating environment, approved services, and selected level. The appendix cases cover all eleven requirement areas: specification; interfaces; roles, services, and authentication; software/firmware security; operating environment; physical security; non-invasive security; sensitive security parameter management; self-tests; life-cycle assurance; and mitigation of other attacks. Separate cases cover Levels 1 through 4. [N13; N14; S2, §29]

A representative case attempts raw key export through a diagnostic interface. The selected module services and the Delegation Broker each apply their rules. Another case withholds signing when a required self-test fails. Laboratory and deployment evidence establish the relevant module properties; the surrounding protocol and resource monitors receive their corresponding evaluation. This gives cryptographic services a defined place within the complete architecture.

7.3. Security categorization and controlled information

FIPS 199 categorization evaluates confidentiality, integrity, and availability consequences for the actual mission and information. The HTTP6 reference architecture connects confidentiality to scoped disclosure and key use, integrity to exact-action binding and admitted state, and availability to bounded failure and controlled recovery supported by capacity and continuity engineering. The

appendix cases represent low, moderate, and high impact separately for each objective. FIPS 200's seventeen minimum security areas receive explicit example references. [E1; E2; S2, §27]

The CUI mapping retains the separate namespaces of all seventeen SP 800-171 Revision 3 families. A reporting-worker example illustrates controlled information flow; a maintenance example constrains technician privileges; a supplier example requires accepted provenance. The governing selection determines applicable requirements and assessment objectives. Agency-selected enhanced requirements receive their own inventory and evidence assignments rather than being inferred from an unrelated control-family label. [E3; E4; S2, §30]

7.4. CSF 2.0 and the lifecycle of responsibility

CSF 2.0 provides six functions through which the same authorization records can support broader organizational outcomes. The appendix catalog retains all twenty-two categories and 106 outcome cases from the expanded mapping. Each case names an event, an expected response, an owner, and required evidence. The following examples illustrate the six-function progression. [N8; S2, §26]

GOVERN connects the human root ceiling to an approved mission, responsible executives, policies, supplier obligations, and risk decisions. A contract change updates the selected requirement inventory. **IDENTIFY** connects admitted machines, software, data, and services to inventories and risk assessments. Discovery of an unmediated route updates both the architecture and the corrective-action record.

PROTECT applies authenticated admission, inherited restrictions, protected data use, training eligibility, and infrastructure safeguards. **DETECT** correlates invalid proof positions, unexpected destinations, changed measurements, and relevant provider events. The monitoring system records observations; authorized analysts decide whether those observations meet incident criteria.

RESPOND uses named lineage to contain the affected worker, parent, or root and to preserve evidence for investigation. **RECOVER** uses approved restoration procedures, verified backups, fresh generations, functional validation, and stakeholder communication. The identity architecture supplies traceable objects across that progression, while the risk program supplies the organizational decisions and operations.

7.5. Lifecycle, privacy, and agent-risk programs

The seven RMF steps—Prepare, Categorize, Select, Implement, Assess, Authorize, and Monitor—organize deployment responsibility. SSDF's four selected Version 1.1 groups organize development evidence: Prepare the Organization, Protect the Software, Produce Well-Secured Software, and Respond to Vulnerabilities. A controlled replacement can preserve an authority ceiling while correcting an implementation defect. [E5; N15; S2, §31]

The selected Privacy Framework Version 1.0 adds Identify-P, Govern-P, Control-P, Communicate-P, and Protect-P to processing involving enrollment and lineage records. The selected AI RMF 1.0 adds GOVERN, MAP, MEASURE, and MANAGE to delegated agent use. The Policy Decision Point constrains an agent's permitted effects; task-specific evaluations address the quality and appropriateness of its outputs. An agent may remain inside its permission while producing an incorrect conclusion, so the evaluation plan preserves both questions. [E11; E12]

7.6. Separate assurance dimensions in the instance profile

Why NIST asks for it. A well-protected workload key answers a different question from the confidence that a particular civil identity was proofed. A machine heartbeat is different from renewed human authentication. A validated cryptographic module has a defined boundary and configuration. Combining these labels into one score would obscure the evidence relevant to each requirement. [N3–N5; N13; N14]

```
assurance_selection:
  identity_profile: organization-selected-IAL
  authentication_profile: organization-selected-AAL
  federation_profile: organization-selected-FAL-if-applicable
  module:
    certificate: null
    exact_version: null
    operational_environment: null
    approved_services: []
  privacy_profile: exact-private-label-or-computational
  human_auth_time: independently-recorded
  machine_liveness: separately-recorded
  decision: pending-deployment-evidence
```

```
CHECK_ASSURANCE_SELECTION(request):
  select clauses separately for proofing, authentication, federation, and modules
  verify each provider's identity, scope, version, and applicable evidence
  require the selected human reauthentication when due
  reject any attempt to substitute workload activity for human presence
```

Appendix I embeds all prior assurance and lifecycle classifications, including IAL/AAL/FAL cases, module requirement areas and levels, CUI families, RMF steps, SSDF groups, privacy functions, and AI-risk functions. The module profile identifies the actual services used by the new review code separately from any future validated production module; testing ECDSA in a Python process is not recorded as module validation. [R3]

8. Worked demonstration: a reporting-agent family

A human approves a root for reading a designated ledger view and publishing approved aggregate reports. A scheduler receives bounded spawning permission. Each worker receives its own protected key, birth record, policy digest, permitted destinations, and twin channel. The ledger, report store, and admission service constitute the approved workflow. The following fixture is condensed from the source examples; its expected outcomes are test criteria. [S2, §§21, 25, 32.5]

```
case_id: reporting-family-demonstration
status: illustrative_not_executed
root: finance-root/e42
worker: report-worker/17
allow:
  operation: report.publish
  destination: reports.example.internal
  data_class: approved-aggregate
challenge_state: unused
cases:
  approved_report: permit_once
  repeated_consumed_proof: no_second_effect
  raw_ledger_export: deny
  locally_broadened_role_catalog: deny_old_admission
  unapproved_child: deny_admission
  retired_parent_epoch: reject_or_expire_within_bound
observed_results: null
```

The positive case confirms that the approved report actually reaches the intended service. The retained evidence connects admission, inherited policy, authenticated destination, sequence, reserved allowance, and report identifier. Testing only denial could conceal a system that blocks all useful work; the authorized path establishes that the controls preserve the specified business function.

The first adverse case repeats the accepted proof through another replica. The resource returns a rejection or the prior transaction result, producing no second effect. The next case substitutes raw ledger export for report publication. Operation, destination, and classification checks reject that request. The assessor then changes the worker's local role definition and repeats the export. The authenticated role snapshot and ancestral ceiling remain controlling.

Additional cases remove an ancestor, forge a child birth, copy the runtime image, change a peer, and race siblings for the final budget allocation. Each case targets a different invariant. The assessor records the gate decision and independently inspects the actual effect, including attempted direct database and unbrokered network paths. A frontend response alone is insufficient evidence of resource enforcement.

Finally, an authorized responder fences the scheduler. Current gates reject the retired ancestry, while lease-based gates expire within the admitted bound. Recovery uses a new generation and retains the earlier transaction outcome. This demonstrates the intended separation between continuing a business task and reviving an obsolete cryptographic identity.

The case can also include a personnel termination or expired hosting assertion. These events enter through their designated trusted services and trigger the response defined in the approved policy. Organizational evidence establishes that the personnel or hosting fact is real and applicable; the runtime evidence establishes how that fact affected authority. [S2, §§25.11, 25.14, 32]

8.1. Inspect the birth, action, and outcome as separate records

The concrete review example contains a root grant signed by a synthetic admission authority, a child grant signed by its parent, a child-signed action, and an independently supplied registry fixture. The packet carries no human secret. The actual secret-to-root transition is represented by the trusted admission boundary, not by a fabricated implementation of `RWVerify`. Appendix F prints the exact public records generated for this edition; no private keys are included.

```
example_records:
  root: examples/review_root_grant.json
  child: examples/review_child_grant.json
  request: examples/review_request.json
  proof: examples/review_action_proof.json
  registry: examples/review_public_registry.json
  code: reference_code/authority_model.py
  status: synthetic-reference-example
```

```
CHECK_REPORT_EXAMPLE:
  verify root with the independently configured admission-authority key
  verify child with its parent key and exact parent-record digest
  check runtime registration, unchanged policy, epoch, and lifetime
  verify the leaf's proof against actual peer, channel, body, and position
  consume the issued position and inspect the durable local effect
```

Why NIST asks for the independent inspection. A successful frontend response does not establish that the database applied the intended restriction, and a denial response does not establish that no side effect occurred. The example checks the local transaction table as well as the gate result. A production test supplies the equivalent resource-side evidence and exercises alternative paths. [N1, AC-25, AU-3; N2]

9. From classification coverage to a complete assessment record

9.1. Define a finite requirement universe

$$\mathscr{R}_{\mathscr{B}} = \bigcup_{d \in \mathscr{D}} \mathsf{Expand}(d, \mathsf{edition}_d, \mathsf{selection}_d, \mathsf{parameters}_d, \mathscr{B}). \tag{9}$$

The system boundary is $\mathscr{B}$ and the selected publication set is $\mathscr{D}$. Expansion produces atomic obligations with stable source locators, including selected enhancements, subordinate clauses, assessment objectives, references, and organization-defined values. The inventory records edition, scope, applicability, and approved selection. It can include a selected CSF outcome with its own success criteria while preserving that outcome's distinct role. [S2, M9]

This supplies a controlled denominator for coverage. An omitted route, unresolved requirement, or accepted risk stays visible in the appropriate record rather than improving the reported percentage by disappearing. SP 800-53B supplies baseline and tailoring context, while official OSCAL and CPRT resources support source-data import. The actual approved inventory determines what complete coverage means for that deployment. [N16; E8; E9]

9.2. Measure assignment and verification separately

$$\begin{aligned} C_{\text{mapped}} &= \frac{|\{r \in \mathscr{R}_{\mathscr{B}} : \mathsf{Assigned}(r)\}|}{|\mathscr{R}_{\mathscr{B}}|}, \\ C_{\text{verified}} &= \frac{|\{r \in \mathscr{R}_{\mathscr{B}} : \mathsf{Closed}(r)\}|}{|\mathscr{R}_{\mathscr{B}}|}. \end{aligned} \tag{10}$$

For a nonempty approved inventory, the first ratio measures complete implementation assignments and the second measures requirements with accepted assessment determinations and current evidence. Classification coverage is tracked separately: it records whether families, functions, or categories appear. An empty or unapproved inventory has undefined coverage. A complete family list therefore provides useful navigation without masquerading as a complete clause assessment. [S2, M10]

The distinction makes work actionable. An access-control requirement can have an assigned broker and owner while bypass testing remains outstanding. The mapped record identifies the intended answer; the verified record identifies the evidence supporting its operation. Reporting both measures preserves progress without turning a design description into a result. This paper supplies classification breadth and expected cases; a deployment fills the requirement-level denominator and observations.

9.3. Close a requirement through its objectives

$$\begin{aligned}\mathsf{Closed}(r) = {} & \mathsf{ScopeApproved}(r) \\ & \wedge \mathsf{ParametersResolved}(r) \\ & \wedge \mathsf{ImplementationBound}(r) \\ & \wedge \mathsf{DependenciesSatisfied}(r) \\ & \wedge \mathsf{EvidenceCurrent}(r) \\ & \wedge \mathsf{ObjectivesAccepted}(r).\end{aligned} \tag{11}$$

A requirement closes when its scope is approved, its parameters are concrete, its implementation is identified, its dependencies are satisfied, its evidence is current, and its applicable objectives have accepted determinations. Examination, interview, and testing are selected according to the objective. Policy documentation, operational instruction, and transaction behavior may require different methods. The record binds each conclusion to the actual deployment or procedure assessed. [S2, M11; N2]

The consequence is a common assurance structure across technical and organizational controls. A replay objective calls for ledger and behavior evidence. A training objective calls for actual delivery and effectiveness records. A physical control calls for applicable facility evidence. The Proof Authority can authenticate the source and scope of supporting records; the reviewer establishes whether they demonstrate the represented safeguard. This preserves the accountability of each provider.

9.4. Detect omissions against an independent inventory

$$\mathscr{U} = \mathscr{R}_{\mathscr{B}} \setminus \mathrm{dom}(\mathscr{M}), \qquad \mathsf{NoOmission} \iff \mathscr{U} = \varnothing. \tag{12}$$

$\mathscr{M}$ maps requirement identifiers to implementation records. Its domain is the set of identifiers present in the register; set difference yields the missing set $\mathscr{U}$. An empty missing set establishes inventory coverage when the independently imported selection is complete and approved. Additional checks reject duplicates, unknown entries, unassigned owners, unresolved parameters, and absent assessment objectives. The original source locator remains attached to each record. [S2, M12]

This prevents a register from declaring itself complete merely by checking its own entries. A selected requirement absent from the mapping is discovered against the external inventory. The supplementary record-completeness checker illustrates this structural discipline with synthetic fixtures. Its role is to identify missing or inconsistent records; assessor review, actual evidence collection, and cryptographic validation establish the substantive determination.

9.5. Carry evidence through dependencies

$$\mathsf{EffectiveEvidence}(r) = \mathsf{OwnEvidence}(r) \wedge \bigwedge_{d \in \mathsf{Dependencies}(r)} \mathsf{ApplicableCurrentEvidence}(d). \tag{13}$$

The requirement's own evidence is combined with the applicable current evidence of every necessary provider. Dependencies identify scope, versions, validity, and accountable owners. An audit receiver's authenticated facility alarm establishes message provenance; installation and inspection records establish the sensor and facility safeguard. A module certificate must apply to the actual module configuration used. Each dependency supplies the evidence for its own proposition. [S2, M13]

The result is accountable inheritance rather than assumed inheritance. A provider with the wrong hosting scope or expired evidence cannot silently support a completed requirement. Dependency cycles require explicit resolution. This permits one well-assessed common service to support several deployments while preserving the conditions under which its evidence is reusable. The same structure applies to proofing, audit retention, training eligibility, and protected execution.

9.6. Reassess the requirements affected by change

$$\Delta \cap \mathsf{DependencyClosure}(r) \neq \varnothing \quad \Longrightarrow \quad \mathsf{Reassess}(r). \tag{14}$$

Δ is the set of changed artifacts, procedures, requirements, or operating conditions. The dependency closure contains the items supporting a requirement's determination. Their intersection identifies affected conclusions. A changed normalizer can affect destination checks; a changed key module can affect several cryptographic services; a new hosting site can affect physical safeguards. Prior evidence remains attached to its historical scope while the successor receives an impact review. [S2, M14]

This connects immutable instance identity to continuous assessment. Controlled replacement creates a new measured generation, and the evidence model identifies the reviews and tests required for that generation. Unchanged applicable evidence can remain useful. Immediate revocation and safety decisions still act when necessary. CA-7 and the RMF Monitor step provide relevant organizational relationships for keeping the implementation and its assurance record aligned. [N1; E5]

9.7. Exchange artifacts and an independently sourced inventory

The appendix dossier identifies a system security plan, assessment plan, assessment results, and corrective-action register as separate review artifacts. Official OSCAL models supply representations for these records, together with catalogs, profiles, and component definitions. The custom architecture records in this edition are proposed mapping objects; an OSCAL export is marked valid only after validation against the exact selected official schema and resolution of its imports. [R2]

```
requirement_assignment:
  source_catalog: independently-obtained-authoritative-edition
  source_clause: exact-control-statement-or-assessment-objective
  implementation: exact-measured-adapter-or-owned-procedure
  parameters: explicitly-resolved
  methods: [examine, interview, test]
  observed_result: null
  evidence: []
  assessment_status: not_assessed
```

```
CHECK_COMPLETENESS(approved_inventory, implementation_register):
  require an independently approved nonempty inventory
  compare identifiers, parameters, objectives, and selected scope
  require current applicable evidence for every dependency
  report missing or incomplete records without generating observations
```

The bulk official catalog is not reproduced as an invented inventory. The package supplies an importer for a reviewer-provided local OSCAL catalog, preserves original source locators and status, and requires approval of the selected scope. The importer produces source inventory, not assessment determinations. Appendix J explains the exchange and selection procedure; Appendix E prints the importer and existing closure-checker source. [E8; E9]

10. Bind the requirement profile to inherited authority

$$h_{\mathscr{R}} = H(\mathsf{Enc}(\mathsf{scope}, \mathsf{editions}, \mathsf{selected_ids}, \mathsf{parameters}))\,,$$
$$D^{+}_{u\to v} = \mathsf{Sign}_{sk_u}\left(\mathsf{Enc}\left(\begin{array}{c} D_{u\to v}, h_{\mathscr{R}}, h_{\mathsf{adapters}}, \\ \mathsf{required_evidence_policy} \end{array}\right)\right). \tag{15}$$

The proposed extension commits the approved boundary, selected editions, requirement identifiers, and resolved parameters to $h_{\mathscr{R}}$. The parent authenticates the ordinary child grant $D_{u\to v}$ together with that profile, the enforcing adapters, and the policy governing evidence dependencies. The signing key is sk_u. Under the signature and hash premises used for delegation, substituting another profile fails authentication. The extension carries forward the source's immutable policy-binding pattern. [S2, M15]

The engineering significance is continuity between approval, execution, and assurance. A child can accept narrower permissions or additional approved prerequisites while retaining the root profile identity. Evidence freshness remains dynamic under the fixed policy. An expired host attestation may hold key use, while a scheduled training-program review may trigger an accountable program action. The approved risk policy determines the response appropriate to each dependency rather than treating every document change as an indiscriminate shutdown condition.

A compact assessment envelope illustrates how the relationship is recorded. It is an assessment extension, not a replacement for the runtime policy schema. The complete raw source format remains in the appendices and accompanying source files. [S2, §24.3]

```
record_kind: illustrative_requirement_assignment
requirement:
  publication: NIST-SP-800-53
  edition: rev5-release-5.2.0
  id: IA-2(8)
implementation:
  owner: identity-service-owner
  mechanism: fresh-context-plus-atomic-consumption
  enforcers: [admission-verifier, resource-gate, ledger]
  deployment_manifest_digest: null
assessment:
  acceptance: one-consumption-per-admitted-epoch-and-nonce
  methods: [examine, test]
  observed: null
  evidence: []
  status: designed_pending_implementation_evidence
```

10.1. Preserve requirement identity without exposing commitment content

The root-profile hash commits to selected obligations, boundary, editions, and parameters. It is deliberately independent of the human secret. A descendant can therefore prove that it inherited the approved assessment profile while the private-reference mechanism protects the original commitment. A change in approved requirements produces an explicit profile transition rather than a hidden mutation of existing authority. [S2, §32.7]

```
profile_transition_example:
  existing_root_profile: reporting-selection-v2
  requested_child_profile: unapproved-selection-v3
  requested_operation: reuse-existing-parent-admission
  expected: deny-profile-substitution
  approved_path: reviewed-new-profile-and-admission-transition
  observed: null
```

```
VERIFY_PROFILE_CONTINUITY(chain):
  obtain the approved profile identity from the trusted root admission
  require each delegation to preserve it and the required adapter binding
```

```
evaluate current evidence only under the pinned evidence policy
route an expanded profile to the authorized approval and replacement workflow
```

Why the pattern matters. A lineage can be cryptographically intact while its software silently uses a different policy interpretation or requirement selection. Explicit profile binding makes that substitution observable. It authenticates what was selected; the evidence register separately establishes whether the actual selected safeguards were implemented and assessed. [N1, CM-3, CA-2; N2]

11. Deployment and evaluation program

The first deployment activity is to define the boundary and approved mission. The owner identifies humans, workloads, agents, protected data, destinations, brokers, resources, and inherited services. Every usable path to an effect receives an enforcement assignment. The requirement inventory pins the selected publications, editions, parameters, and assessment objectives. The same inventory drives both the implementation plan and the evidence plan. [S2, §§23–24, 32]

The second activity is to qualify the admission chain. Enrollment and the selected identity profile establish the human relationship. Protected launch and attestation bind each instance key to the expected runtime. The issuance broker verifies templates, roles, ancestry, budgets, and current generation before a child leaves quarantine. The service twin confirms the admitted peer, policy, epoch, and sequence. Operational owners establish recovery and revocation procedures before broadening deployment.

The acceptance campaign pairs useful authorized work with adverse cases. Its principal cases are replay races, local policy mutation, ancestor removal, copied instances, destination substitution, shared-budget races, human-session expiration, parent fencing, recovery rollback, and stale provider evidence. The resource's actual outcome is inspected separately from the requesting application's response. Test records identify versions, environment, time, expected behavior, observed behavior, and reviewer. [S2, §32.5]

Operational exercises complement the software tests. A training record is backed by delivered instruction. A maintenance ticket is backed by authorized servicing. A media-disposition record is backed by verified sanitization. A facility record is backed by inspection and exercise. A supplier approval is backed by reviewed deliverables. These activities establish the represented facts, while the Policy Decision Point supplies the controlled relationship between those facts and digital authority.

Evaluation reports identify selected requirements, mapped assignments, unresolved parameters, assessment objectives, dependencies, and findings. Technical observations include duplicate effects, unauthorized effects in the campaign, cessation intervals, and independently enforced resource paths. Formal propositions and finite observations retain their respective premises and test conditions. These records define a reproducible review program. [S2, §§21, 32.6]

11.1. Review deliverables and reproducible execution

Why NIST asks for this material. Reviewers need the implementation statement, methods, scope, and evidence needed to reproduce the relevant conclusions. No single generic attachment list establishes every NIST program's acceptance criteria. The dossier in Appendix J makes a concrete architecture-review package and assigns the deployment-specific proofs, certificates, and operational records to their responsible owners. [N2; E5; R2]

```
review_package:
  front_matter: [abstract, introduction, reading-guide]
  models: [observer-view, lifecycle-state-machine, finite-invariant-model]
  schemas: [original-policy, review-objects, protocol-contracts]
  algorithms: [ignite, mint, verify-chain, authorize-effect, recover]
  examples: [signed-public-records, twenty-control-families, csf-106]
  code: [authority-model, privacy-model, state-model, catalog-importer, closure-checker]
  evidence: [local-test-records, schema-checks, model-results, source-digests]
  deployment_evidence: assigned-separately-to-system-owners
```

```
BUILD_REVIEW_EDITION:
  preserve supplied sources and their hashes
  assemble chapters and embedded appendices in a fixed order
  validate schemas and examples; execute bounded reference tests
```

```
record actual results with scope and environment
typeset, inspect, and publish the document with the editable source package
```

The test results in Appendix K are local reference-model observations. Genuine private-reference security evidence, real attestation validation, distributed ledger tests, resource-bypass testing, and operational assessments remain distinct records. This edition provides the executable starting point and a review index without assigning invented production outcomes.

12. Conclusion

The HTTP6 reference architecture connects the beginning of authority to its continuing exercise. A human approves a bounded root; distinct children inherit narrower grants; protected resources verify current identity, policy, state, and an action-specific proof. The model makes accountability part of execution rather than a label attached only to the initial login.

Its formal consequences are specific and consequential. Atomic consumption supplies one-use authorization. Policy intersection preserves the root ceiling. Authenticated commitments preserve admitted policy meaning. Shared accounting conserves family budgets. Epochs and leases make cessation measurable. Conditional transcript independence specifies a privacy result for a defined observer and randomness model. These properties organize a coherent response to several NIST authentication, access, integrity, and accountability objectives.

The wider contribution is the connection between authority and evidence. Technical mechanisms, integrated services, inherited safeguards, and organizational operations receive explicit owners and assessment paths. The requirement inventory defines the scope; the register exposes omissions; dependency-aware evidence establishes support; change review keeps conclusions aligned with the actual deployment. A classification becomes a concrete example, an example becomes an assessment objective, and the implemented objective becomes reviewable evidence.

Human approvals remain attached to their authorized effects. Machine instances maintain their individual right to act, and agents exercise only independently enforced grants. The human pilot light, hereditary policy, and evidence model together define an architecture for accountable, NIST-aligned automation.

12.1. An explicit public claim and an explicit review path

The architecture's public claim is strongest when each element states its mechanism. The original human commitment is not a field required in standard child delegation. Under the binary-conditioned sequential-independence proposition, even a trillion observed successful ceremonies adds no information about that commitment. Under authentic admission, secure delegation, correct state, and complete mediation, child authority remains traceable to the human-originated root and bounded by its ceiling. These are complementary propositions, not one property inferred from the other. [S1; D1]

The practical answer to NIST objectives is consequently a reviewable design: an authorized mint creates distinct descendants, committed policy fixes their meaning, fresh proofs bind their actions, and resource gates preserve the decision through execution. Examples show why those safeguards are needed. Schemas identify their records. Algorithms identify their enforcement sequence. Source models and tests make selected mechanisms executable. A requirement register assigns the remaining implementation and operational evidence to actual owners.

```
REVIEW_CONCLUSION(claim):
  identify which proposition or requirement is being asserted
  inspect its complete inputs, trust boundary, and implementation
  verify the matching evidence at the declared scope
  publish only the conclusion supported by that combined record
```

Hereditary authority can therefore scale the human decision without copying the human commitment into every machine. The proposed Rosario–Wang admission relation supplies a private origin; the Minting and Delegation Service carries the restricted authorization through individually accountable circuit instances. This edition presents both the argument and the artifacts through which an independent reviewer can examine that relationship.

Appendix A. Review model, trust boundaries, and implementation contracts

A.1. Objects and terminology

This edition uses **commitment** in the source's sense of the human's enrolled secret relationship. Let S denote its secret content and E the protected enrollment representation. A public cryptographic commitment, if a deployment introduces one, is a separately specified object $C = \mathsf{Commit}(S; r)$ with its own hiding, binding, and randomness analysis. A plain digest of a guessable secret is not used here as a synonym for a proven hiding commitment. Policy commitments are hashes of canonical policy bytes, not hashes of the human secret.

A **binary artifact** B is the exact acquired byte sequence in the stated possession experiment, plus any extracted contents the experiment allows. A **runtime instance** v is a particular admitted execution with a protected key, birth identifier, policy digest, epoch, and appraised machine binding. A **workload** is a logical service that can have many such instances. A **grant** is authenticated authorization to an identified instance. A **mint** is the protected function that creates grants after policy and state checks. A **twin** is an authenticated peer maintaining corresponding logical state. None of these terms implies physical quantum entanglement. [S1]

The main paper retains Equations 1–15. New labels beginning with A, B, or D identify this edition's review definitions and deductions. The runnable code is an original reference model for this edition, not a recovered copy of proprietary Rosario–Wang implementation code. The source provides the verification interface and conditional propositions; a production submission supplies the actual enrolled-relation implementation, private-map lifecycle, hardware binding, and protocol test vectors. [S1; D1; D2]

A.2. Five security boundaries

The human interaction boundary contains trusted presentation, input, private-reference handling, and the transaction the person is approving. The admission boundary authenticates that outcome and binds it to a root key and policy. The minting boundary appraises the proposed child and restricts its grant. The execution boundary isolates the admitted instance and its key-use operations. The resource boundary authorizes the actual database operation, decryption, network transfer, or agent tool effect. Audit and evidence services observe these boundaries through separately authorized interfaces.

The review assumptions identify who controls each boundary and how compromise changes the claim. A file copied from storage supplies bytes to the observer; it does not establish possession of a non-exportable runtime key. Conversely, a test that actually extracts a key includes that key in the attacker's state. The review never treats exposed contents as private merely because they originally lived in a circuit. Claims for the resulting attack model are then evaluated anew.

A.3. Artifact-possession and adversary matrix

Review experiment	Observer receives	Property evaluated
AP-1: executable acquisition	Complete code and inspectable configuration	Incremental secret privacy conditioned on those bytes
AP-2: enrolled binary acquisition	AP-1 plus every serialized map, seed, enrollment field, or key actually present	Same privacy test with the larger known view; identify any offline verifier

Review experiment	Observer receives	Property evaluated
AP-3: passive history	Binary plus challenge, display, witness, result, and admitted timing history	Joint sequential transcript independence or computational privacy
AP-4: active protocol use	AP-3 plus chosen requests, rejection responses, retries, and state resets permitted by the environment	Soundness, rate limiting, state consistency, and active leakage
AP-5: copied running image	Snapshot bytes and resumed software	Key custody, generation fencing, fork resistance, and current registration
AP-6: privileged runtime exposure	Whatever protected material the exploit actually obtains	Explicit post-exposure scope and authenticated recovery
AP-7: colluding children	All authorized child grants, local keys, and observations in the collusion	Ancestor ceilings, scoped keys, and shared-budget conservation
AP-8: infrastructure interruption	Packet loss, replica partition, crashes, clock skew within declared limits	Safe failure, durable consumption, and bounded recovery

Privacy and authorization experiments retain separate outputs. A circuit can conceal S yet expose a signing key that permits forgery. A circuit can preserve a signing key while a badly designed transcript reveals part of S. Both outcomes must be captured in the review record rather than inferred from each other.

A.4. Protocol role contracts

`RWVerify` accepts an enrollment handle, a challenged geometry, a witness sequence, and an authenticated context. It returns a single admission result plus an opaque receipt handle. Its concrete contract specifies completeness, soundness, allowed observer view, state use, error behavior, and enrollment/recovery transitions. It supplies no public export of the original secret. A reviewer examines the actual implementation and its corresponding argument before assigning a production assurance result.

`Attest` supplies evidence for an expected key, runtime measurement, challenge, machine generation, and policy binding. `Appraise` checks that evidence against independently maintained trust roots and reference values. `Mint` accepts an appraised child and a live parent, then emits a policy-bounded grant. `VerifyChain` authenticates every link to a recognized human-admission root. `AuthorizeEffect` evaluates the complete request at the resource and creates its durable one-use decision.

`Canonicalize` fixes byte-level meaning. The production profile identifies its exact standard and domain-specific normalization rules. The runnable model uses the deliberately narrower `TEST-CANON-1`: printable ASCII strings, null, Boolean values, safe integers, lists, and string-keyed objects; sorted keys and compact JSON delimiters. It rejects floats and unsupported Unicode. This is a deterministic test encoding, not a claim to implement every JSON Canonicalization Scheme rule. [R1]

A.5. Protected source code submission inventory

The review receives the production sources for enrollment, challenge generation, private-label construction, proof checking, key-use authorization, minting, chain validation, normalization, resource adapters, state storage, recovery, and audit. It also receives build scripts, dependency locks, compiler settings, release measurements, test vectors, and the mapping from source to installed artifacts. Public disclosure of the human commitment is unnecessary for this review; synthetic enrollment data supplies reproducible fixtures, while confidential implementation material can follow the reviewer's approved handling process.

This edition supplies the architecture contracts, full existing policy schema, a separate executable review-object schema, reference implementations for delegation and consumption, finite privacy and state models, the existing record-completeness checker, and their tests. The production Rosario–Wang engine, genuine hardware-attestation adapter, deployed consensus ledger, and independently collected field evidence are named integration deliverables. Their absence from the supplied manuscripts is recorded as a deliverable assignment, not replaced with invented source or a passing result.

A.6. Functional and security acceptance

A valid positive test exercises the intended business action through the same enforcement path used by negative tests. A negative test specifies the changed field, expected gate, and independently inspected effect. An application response alone is not the effect inspection. Every execution trace identifies code version, policy, instance, root, epoch, peer, request digest, decision, consumption position, outcome, and collector. Sensitive private references and live keys are excluded from ordinary evidence exports.

The evaluator distinguishes proof premises, finite model observations, source-code tests, integrated deployment tests, and organizational examination. That separation makes the package useful for independent review: each assertion has an identifiable supporting object and a clearly bounded interpretation. SP 800-53A supplies the assessment-method context; the selected system boundary and program determine which particular deliverables are mandatory. [N2; E5; R2]

Appendix B. Binary-possession privacy and hereditary authority propositions

B.1. The complete observer view

Let S denote the original human commitment, B all acquired circuit bytes, and P_0 the remaining initial public information. Let O_τ include the entire additional observation of ceremony τ: challenge, exposed geometry, messages, reported witnesses, success status, admitted timing, and any visible delegation fields. An honest-success-only experiment and an active-query experiment are distinct profiles. A proof for one is identified as such. [S1, II.4; D1]

$$V_0 = (B, P_0), \qquad V_\tau = (V_0, O_1, \dots, O_\tau). \tag{B1}$$

The tuple V_0 is the attacker's baseline knowledge. The later view adds complete observations without removing information already known. "Possession of the binary" is therefore part of the mathematical argument rather than an informal exception. A disclosed seed, map, or verification table appears in B whenever the acquisition experiment reveals it.

This formulation connects the source's conditional privacy proposition to an explicit review experiment. It also keeps transcript privacy separate from whether the binary alone exposes secret information. The requested additional-information result compares the posterior after observation with the prior after acquiring that actual binary.

B.2. Sequential independence theorem

Proposition B1 — zero incremental information under binary possession. For every supported history and each ceremony, suppose the complete next observation is conditionally independent of S given the preceding view. Then, for every finite supported n:

$$\begin{aligned} &O_\tau \perp S \mid (V_0, O_{<\tau}) \quad (1 \le \tau \le n), \\ &I(S; O_{1:n} \mid V_0) = \sum_{\tau=1}^{n} I(S; O_\tau \mid V_0, O_{<\tau}) = 0. \end{aligned} \tag{B2}$$

I is conditional mutual information and $\perp$ denotes conditional independence. The chain rule expands the history into increments. Each increment is zero by the premise, so their sum is zero. This proves the result for $n = 10^{12}$ and any other finite n for which the profile supplies the stated conditions. The proof uses no experimental extrapolation from a smaller count.

The resulting public statement is exact: **under this profile, even an observer possessing the specified binary gains no additional information about the original commitment from a trillion successful-login observations.** Establishing that an actual Rosario–Wang implementation satisfies the premise requires its complete construction, view specification, and exposure analysis. The source supplies the conditional theorem and an ideal sufficient masking construction; it does not supply an unconditional proof for every fully exposed, self-contained binary.

For supported events with nonzero probability, the equivalent posterior statement is:

$$\Pr[S = s \mid V_0 = v, O_{1:n} = o] = \Pr[S = s \mid V_0 = v]. \tag{B3}$$

The probability of each candidate secret remains what it was after the initial acquisition. A nonuniform human-secret distribution remains nonuniform; the theorem preserves uncertainty rather than manufacturing a uniform prior. No statistical estimator gains information from those observations under the proposition's premises.

A stronger total-disclosure claim additionally requires $I(S; V_0) = 0$. By the chain rule, baseline secrecy and zero incremental leakage together give $I(S; V_0, O_{1:n}) = 0$. Reviewers must check both when the public claim concerns the binary itself as well as the observation history. If the baseline already contains the secret, zero incremental leakage alone is vacuous as protection of that secret.

B.3. A sufficient exact private-label realization

For L membership positions and m response labels, let X_τ be the underlying membership-index vector. Provide an independent uniform private vector Z_τ that remains outside V_0 and the observable history. Report:

$$\begin{aligned} W_\tau &= (X_\tau + Z_\tau) \bmod m, \\ \Pr[W_\tau = w \mid S = s, V_0, O_{<\tau}] &= m^{-L}. \end{aligned} \tag{B4}$$

Addition and remainder are coordinatewise. For any fixed underlying vector and desired witness, one mask produces that witness; every mask has probability m^{-L}. The witness distribution is therefore identical across secret choices. An implementation also ensures that every other component of O_τ preserves the same independence; visible private labels, secret-dependent diagnostics, or distinguishing acceptance behavior receive explicit analysis rather than being omitted.

The protected interpretation can be provided by an independently protected user/verifier component or another precisely specified mechanism. The review records how the human and verifier obtain the relevant interpretation, how it is renewed, and what an observer holding the binary can recover. This ideal model is a sufficient privacy example, not a substitution for the actual Rosario–Wang algorithm or a complete authentication protocol. Soundness and authorization still require their own predicates.

B.4. Provisioning a trillion-observation exact profile

For the independent-mask construction, deterministic generation from all private mask material K obeys:

$$H(Z_{1:n} \mid V_0) = nL \log_2 m \le H(K \mid V_0). \tag{B5}$$

H denotes Shannon entropy. A deterministic transformation cannot create additional conditional entropy. With $n = 10^{12}$, $L = 20$, and $m = 3$, the independent masks contain approximately 3.17×10^{13} bits, or 3.96 trillion bytes of entropy. Storage encoding and secure distribution have their own overheads. A 65,536-bit pool dedicated entirely to independent masks supplies only about 2,067 such twenty-position ceremonies before replenishment in this particular profile.

Thus the exact profile provisions, distributes, and consumes fresh private randomness over the required lifetime; the computational profile uses pseudorandom expansion under specified assumptions. A timestamp supplies an index, not secret entropy. The bound belongs to this independent-mask realization and is not asserted as a universal lower bound on every possible zero-knowledge or authentication protocol. An alternative Rosario–Wang proof can establish its own construction-specific resource and composition argument. [S1, II.4.3; N10–N12]

B.5. Exposed deterministic interpretation as a diagnostic case

The complete-view model gives a useful check on proposed implementations. If the witness is a deterministic, nonconstant function of S and fully known circuit/context, then:

$$H(W \mid S, V_0) = 0 \quad \Longrightarrow \quad I(S; W \mid V_0) = H(W \mid V_0). \tag{B6}$$

The identity follows from the definition of conditional mutual information. Whenever the witness has positive entropy under the known view, it conveys positive information about the secret. For example, a known first partition separates four possible symbols into two pairs; a known second partition separates the remaining pair. Observing both plain membership bits identifies the symbol. The finite source example in Appendix E checks this case and the fresh-private-mask contrast exactly.

This diagnostic identifies the engineering condition that must change: the observation model must retain an independently private interpretation or the construction must provide a different formally established privacy mechanism. The review includes every exposed byte instead of claiming perfect secrecy from balanced regions alone. A static private permutation reused indefinitely also requires its own repeated-history analysis; independent-looking marginal responses do not establish joint-history independence.

B.6. Secrecy and heredity compose through separate mechanisms

Let R_h be an authenticated human-admission receipt that binds the approved root key, policy, and epoch without serializing S. Let $D_{u \to v}$ be a signed grant to child v, and let each effective policy be the intersection of its ancestry. Under authentic admission, unforgeable signatures, collision-resistant commitments, correct runtime binding, and complete mediation:

$$\mathsf{AcceptedEffect}(v, q) \Longrightarrow \mathsf{Anchored}(v, R_h) \wedge q \in \mathscr{A}_0. \tag{B7}$$

The first term follows by validating each parent link until the trusted admission receipt; the second follows by induction over permission intersections. No child needs the human's secret to establish either term. A certificate containing a claim of ancestry is accepted only when the links, signer authority, key possession, current state, and resource policy validate.

The architectural conclusion is that private-reference admission is a strong candidate for hereditary authorization because the secret can remain at the origin while permission is independently propagated and verified. Transcript privacy protects the origin; cryptographic delegation and policy attenuation protect the descendants. Mint permission is explicit and bounded. Public provenance may identify an authorization lineage, so anonymity and unlinkability of people or tasks are separately selected privacy goals rather than consequences of secret non-disclosure.

B.7. Required proof and evidence deliverables

The construction dossier supplies the enrollment algorithm, exact witness relation, private-map encoding, every binary field, randomness source and budget, the full observable transcript, active-query rules, soundness bound, sequential composition argument, erasure behavior, and recovery transitions. An artifact-exposure test inventories what acquisition actually reveals. A privacy analysis proves the relevant distributional or simulation statement. Finite enumeration verifies small instances, while implementation tests check that serialized outputs match the specified view. None of those finite checks is described as execution of a trillion logins.

A reviewer also checks denial-of-service and guess attempts separately. A privacy-preserving transcript can coexist with a weak online guessing probability; a correct cryptographic receipt can coexist with a missing policy gate. The main paper’s rate limits, device binding, current epochs, and action controls supply the corresponding mechanisms. The package records each result under its own claim identifier so that success on one question never silently supplies evidence for another.

Appendix C. Complete schemas and typed data structures

The original `http6.policy.v1` schema is retained from the hereditary-authority package. Its full source follows. The new review-object schema is a separate, narrower executable profile; the proposed protocol-contract schema describes integration records for human admission, appraisal, requirement selection, audit, and provider evidence. These are HTTP6 proposals rather than official NIST schemas. Structural validation is followed by semantic, signature, and authoritative-state checks. [S1; D2]

A signature field contains evidence to verify, not permission to trust the string. A self-asserted `accepted` or `approved` field is authoritative only after the named issuer, signature, scope, freshness, and applicable policy are verified. Cross-field conditions such as child lifetime, ancestry, and profile equality are checked by the algorithms in D and E.

C.1. Authority_Policy_Schema.json

Source path: schemas/Authority_Policy_Schema.json.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:http6:policy-schema:v1",
  "title": "HTTP6 immutable policy content - proposed v1 profile",
  "description": "Structural schema for immutable policy content. A signed instance manifest binds this
  ↪  content by canonical digest. Admission additionally verifies signatures, reference authenticity,
  ↪  cross-field semantics, runtime evidence, ancestor conjunction, and authoritative state. Example
  ↪  values do not constitute an admitted deployment.",
  "type": "object",
  "properties": {
    "schema_version": {
      "const": "http6.policy.v1"
    },
    "policy_id": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048
    },
    "semantics": {
      "type": "object",
      "properties": {
        "profile": {
          "const": "http6-monotone-v1"
        },
        "default_decision": {
          "const": "deny"
        },
        "conflict_resolution": {
          "const": "deny_overrides"
        },
        "evaluation_error": {
          "const": "deny"
        },
        "ancestor_combiner": {
          "const": "conjunction"
        },
        "canonicalization": {
          "const": "RFC8785"
        },
        "engine_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "normalizer_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "action_registry_digest": {
```

```
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "context_resolver_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "operation_adapter_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        }
      },
      "required": [
        "profile",
        "default_decision",
        "conflict_resolution",
        "evaluation_error",
        "ancestor_combiner",
        "canonicalization",
        "engine_digest",
        "normalizer_digest",
        "action_registry_digest",
        "context_resolver_digest",
        "operation_adapter_digest"
      ],
      "additionalProperties": false
    },
    "iam": {
      "type": "object",
      "properties": {
        "tenant": {
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "active_roles": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 64,
          "uniqueItems": true
        },
        "role_catalog_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "assignment_snapshot_digest": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "role_grants": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "role": {
                "type": "string",
                "minLength": 1,
                "maxLength": 2048
              },
              "operations": {
                "type": "array",
                "items": {
                  "type": "string",
                  "minLength": 1,
                  "maxLength": 2048
                },
```

```
              "minItems": 1,
              "maxItems": 128,
              "uniqueItems": true
            }
          },
          "required": [
            "role",
            "operations"
          ],
          "additionalProperties": false
        },
        "minItems": 1,
        "maxItems": 64,
        "uniqueItems": true
      }
    },
    "required": [
      "tenant",
      "active_roles",
      "role_catalog_digest",
      "assignment_snapshot_digest",
      "role_grants"
    ],
    "additionalProperties": false
  },
  "permissions": {
    "type": "object",
    "properties": {
      "allow": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/allow_rule"
        },
        "minItems": 0,
        "maxItems": 256,
        "uniqueItems": true
      },
      "deny": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/deny_rule"
        },
        "minItems": 0,
        "maxItems": 256,
        "uniqueItems": true
      }
    },
    "required": [
      "allow",
      "deny"
    ],
    "additionalProperties": false
  },
  "network": {
    "type": "object",
    "properties": {
      "broker_only": {
        "const": true
      },
      "reauthorize_redirects": {
        "const": true
      },
      "trust_bundle_digest": {
        "type": "string",
        "pattern": "^sha256:[0-9a-f]{64}$"
      },
      "source_route_id": {
        "type": "string",
        "minLength": 1,
        "maxLength": 2048
      },
```

```
        "allow": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/destination"
          },
          "minItems": 0,
          "maxItems": 128,
          "uniqueItems": true
        },
        "deny": {
          "type": "object",
          "properties": {
            "dns_names": {
              "type": "array",
              "items": {
                "type": "string",
                "minLength": 1,
                "maxLength": 2048
              },
              "minItems": 0,
              "maxItems": 128,
              "uniqueItems": true
            },
            "service_identities": {
              "type": "array",
              "items": {
                "type": "string",
                "minLength": 1,
                "maxLength": 2048
              },
              "minItems": 0,
              "maxItems": 128,
              "uniqueItems": true
            },
            "ip_cidrs": {
              "type": "array",
              "items": {
                "type": "string",
                "minLength": 1,
                "maxLength": 2048
              },
              "minItems": 0,
              "maxItems": 128,
              "uniqueItems": true
            }
          },
          "required": [
            "dns_names",
            "service_identities",
            "ip_cidrs"
          ],
          "additionalProperties": false
        }
      },
      "required": [
        "broker_only",
        "reauthorize_redirects",
        "trust_bundle_digest",
        "source_route_id",
        "allow",
        "deny"
      ],
      "additionalProperties": false
    },
    "delegation": {
      "type": "object",
      "properties": {
        "enabled": {
          "type": "boolean"
        },
        "max_depth": {
```

```
          "type": "integer",
          "minimum": 0,
          "maximum": 64
        },
        "max_direct_children": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "max_total_descendants": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "child_template_digests": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern": "^sha256:[0-9a-f]{64}$"
          },
          "minItems": 0,
          "maxItems": 128,
          "uniqueItems": true
        }
      },
      "required": [
        "enabled",
        "max_depth",
        "max_direct_children",
        "max_total_descendants",
        "child_template_digests"
      ],
      "additionalProperties": false,
      "allOf": [
        {
          "if": {
            "properties": {
              "enabled": {
                "const": false
              }
            }
          },
          "then": {
            "properties": {
              "max_depth": {
                "const": 0
              },
              "max_direct_children": {
                "const": 0
              },
              "max_total_descendants": {
                "const": 0
              },
              "child_template_digests": {
                "maxItems": 0
              }
            }
          }
        },
        {
          "if": {
            "properties": {
              "enabled": {
                "const": true
              }
            }
          },
          "then": {
            "properties": {
              "max_depth": {
                "minimum": 1
```

```
            },
            "max_direct_children": {
              "minimum": 1
            },
            "max_total_descendants": {
              "minimum": 1
            },
            "child_template_digests": {
              "minItems": 1
            }
          }
        }
      }
    ]
  },
  "lifecycle": {
    "type": "object",
    "properties": {
      "root_restart": {
        "const": "human_proof"
      },
      "child_restart": {
        "const": "live_parent_fresh_attestation"
      },
      "route_change": {
        "const": "new_admission"
      },
      "lease_seconds": {
        "type": "integer",
        "minimum": 1,
        "maximum": 9007199254740991
      },
      "max_lifetime_seconds": {
        "type": "integer",
        "minimum": 1,
        "maximum": 9007199254740991
      }
    },
    "required": [
      "root_restart",
      "child_restart",
      "route_change",
      "lease_seconds",
      "max_lifetime_seconds"
    ],
    "additionalProperties": false
  },
  "runtime": {
    "type": "object",
    "properties": {
      "launcher_digest": {
        "type": "string",
        "pattern": "^sha256:[0-9a-f]{64}$"
      },
      "enforcement_profile_digest": {
        "type": "string",
        "pattern": "^sha256:[0-9a-f]{64}$"
      },
      "approved_code_digests": {
        "type": "array",
        "items": {
          "type": "string",
          "pattern": "^sha256:[0-9a-f]{64}$"
        },
        "minItems": 1,
        "maxItems": 128,
        "uniqueItems": true
      },
      "machine_inventory_digest": {
        "type": "string",
        "pattern": "^sha256:[0-9a-f]{64}$"
```

```
        },
        "require_fresh_attestation": {
          "const": true
        },
        "require_nonexportable_instance_key": {
          "const": true
        },
        "inherit_instance_keys": {
          "const": false
        },
        "plugin_digests": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern": "^sha256:[0-9a-f]{64}$"
          },
          "minItems": 0,
          "maxItems": 128,
          "uniqueItems": true
        }
      },
      "required": [
        "launcher_digest",
        "enforcement_profile_digest",
        "approved_code_digests",
        "machine_inventory_digest",
        "require_fresh_attestation",
        "require_nonexportable_instance_key",
        "inherit_instance_keys",
        "plugin_digests"
      ],
      "additionalProperties": false
    },
    "proof": {
      "type": "object",
      "properties": {
        "profile": {
          "const": "http6-computational-v1"
        },
        "action_body_bound": {
          "const": true
        },
        "one_use": {
          "const": true
        },
        "channel_bound": {
          "const": true
        },
        "strict_sequence": {
          "const": true
        },
        "ledger_namespace": {
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        }
      },
      "required": [
        "profile",
        "action_body_bound",
        "one_use",
        "channel_bound",
        "strict_sequence",
        "ledger_namespace"
      ],
      "additionalProperties": false
    },
    "budgets": {
      "type": "object",
      "properties": {
        "ledger_id": {
```

```
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "scope": {
          "const": "subtree"
        },
        "limits": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "dimension": {
                "enum": [
                  "operations",
                  "export_bytes",
                  "spend_minor",
                  "child_births",
                  "concurrent_workers"
                ]
              },
              "maximum": {
                "type": "integer",
                "minimum": 0,
                "maximum": 9007199254740991
              },
              "window_seconds": {
                "type": "integer",
                "minimum": 1,
                "maximum": 9007199254740991
              }
            },
            "required": [
              "dimension",
              "maximum",
              "window_seconds"
            ],
            "additionalProperties": false
          },
          "minItems": 0,
          "maxItems": 16,
          "uniqueItems": true
        }
      },
      "required": [
        "ledger_id",
        "scope",
        "limits"
      ],
      "additionalProperties": false
    },
    "audit": {
      "type": "object",
      "properties": {
        "event_types": {
          "type": "array",
          "items": {
            "enum": [
              "ignition",
              "birth",
              "decision",
              "outcome",
              "revoke",
              "recovery",
              "rejection"
            ]
          },
          "minItems": 1,
          "maxItems": 7,
          "uniqueItems": true
        },
```

```
        "private_reference_logging": {
          "const": false
        },
        "retention_seconds": {
          "type": "integer",
          "minimum": 1,
          "maximum": 9007199254740991
        }
      },
      "required": [
        "event_types",
        "private_reference_logging",
        "retention_seconds"
      ],
      "additionalProperties": false
    }
  },
  "required": [
    "schema_version",
    "policy_id",
    "semantics",
    "iam",
    "permissions",
    "network",
    "delegation",
    "lifecycle",
    "runtime",
    "proof",
    "budgets",
    "audit"
  ],
  "additionalProperties": false,
  "$defs": {
    "condition": {
      "oneOf": [
        {
          "type": "object",
          "properties": {
            "field": {
              "enum": [
                "request.tenant",
                "request.amount_minor",
                "request.data_classification",
                "request.method",
                "request.tool",
                "request.recipient",
                "resource.owner",
                "resource.classification",
                "context.region",
                "context.attested",
                "context.source_zone",
                "context.purpose"
              ]
            },
            "operator": {
              "enum": [
                "eq",
                "ne",
                "in",
                "not_in",
                "lte",
                "gte"
              ]
            },
            "value": {
              "oneOf": [
                {
                  "type": [
                    "string",
                    "integer",
                    "boolean"
```

```
            ],
            "maxLength": 2048,
            "minimum": -9007199254740991,
            "maximum": 9007199254740991
          },
          {
            "type": "array",
            "items": {
              "type": [
                "string",
                "integer",
                "boolean"
              ],
              "maxLength": 2048,
              "minimum": -9007199254740991,
              "maximum": 9007199254740991
            },
            "minItems": 1,
            "maxItems": 256,
            "uniqueItems": true
          }
        ]
      }
    },
    "required": [
      "field",
      "operator",
      "value"
    ],
    "additionalProperties": false,
    "allOf": [
      {
        "if": {
          "properties": {
            "operator": {
              "enum": [
                "in",
                "not_in"
              ]
            }
          }
        },
        "then": {
          "properties": {
            "value": {
              "type": "array",
              "items": {
                "type": [
                  "string",
                  "integer",
                  "boolean"
                ],
                "maxLength": 2048,
                "minimum": -9007199254740991,
                "maximum": 9007199254740991
              },
              "minItems": 1,
              "maxItems": 256,
              "uniqueItems": true
            }
          }
        },
        "else": {
          "properties": {
            "value": {
              "type": [
                "string",
                "integer",
                "boolean"
              ],
              "maxLength": 2048,
```

```
                    "minimum": -9007199254740991,
                    "maximum": 9007199254740991
                  }
                }
              }
            },
            {
              "if": {
                "properties": {
                  "operator": {
                    "enum": [
                      "lte",
                      "gte"
                    ]
                  }
                }
              },
              "then": {
                "properties": {
                  "value": {
                    "type": "integer",
                    "minimum": -9007199254740991,
                    "maximum": 9007199254740991
                  }
                }
              }
            }
          ]
        },
        {
          "type": "object",
          "properties": {
            "all": {
              "type": "array",
              "items": {
                "$ref": "#/$defs/condition"
              },
              "minItems": 1,
              "maxItems": 64,
              "uniqueItems": true
            }
          },
          "required": [
            "all"
          ],
          "additionalProperties": false
        },
        {
          "type": "object",
          "properties": {
            "any": {
              "type": "array",
              "items": {
                "$ref": "#/$defs/condition"
              },
              "minItems": 1,
              "maxItems": 64,
              "uniqueItems": true
            }
          },
          "required": [
            "any"
          ],
          "additionalProperties": false
        }
      ]
    },
    "allow_rule": {
      "type": "object",
      "properties": {
        "id": {
```

```
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "operations": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 128,
            "pattern": "^[a-z][a-z0-9_.:-]*$"
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "destinations": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "resources": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "conditions": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/condition"
          },
          "minItems": 0,
          "maxItems": 64,
          "uniqueItems": true
        }
      },
      "required": [
        "id",
        "operations",
        "destinations",
        "resources",
        "conditions"
      ],
      "additionalProperties": false
    },
    "deny_rule": {
      "type": "object",
      "properties": {
        "id": {
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "operations": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 128,
```

```
            "pattern": "^(?:\\*|[a-z][a-z0-9_.:-]*)$"
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "destinations": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "resources": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 256,
          "uniqueItems": true
        },
        "conditions": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/condition"
          },
          "minItems": 0,
          "maxItems": 64,
          "uniqueItems": true
        }
      },
      "required": [
        "id",
        "operations",
        "destinations",
        "resources",
        "conditions"
      ],
      "additionalProperties": false
    },
    "destination": {
      "type": "object",
      "properties": {
        "id": {
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "dns_name": {
          "type": "string",
          "minLength": 1,
          "maxLength": 253,
          "pattern": "^[a-z0-9](?:[a-z0-9.-]*[a-z0-9])?$"
        },
        "service_identity": {
          "type": "string",
          "minLength": 1,
          "maxLength": 2048
        },
        "ports": {
          "type": "array",
          "items": {
            "type": "integer",
            "minimum": 1,
```

```
            "maximum": 65535
          },
          "minItems": 1,
          "maxItems": 64,
          "uniqueItems": true
        },
        "protocols": {
          "type": "array",
          "items": {
            "enum": [
              "tls",
              "quic"
            ]
          },
          "minItems": 1,
          "maxItems": 2,
          "uniqueItems": true
        },
        "ip_cidrs": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "maxLength": 2048
          },
          "minItems": 1,
          "maxItems": 64,
          "uniqueItems": true
        }
      },
      "required": [
        "id",
        "dns_name",
        "service_identity",
        "ports",
        "protocols",
        "ip_cidrs"
      ],
      "additionalProperties": false
    }
  }
}
```

C.2. protocol_contracts.schema.json

Source path: schemas/protocol_contracts.schema.json.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:http6:proposed-protocol-contracts:v1",
  "title": "Proposed production integration record contracts, not a standardized wire format",
  "$defs": {
    "signature": {
      "type": "object",
      "properties": {
        "algorithm": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "issuer_key_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "signature": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        }
      },
```

```
      "required": [
        "algorithm",
        "issuer_key_id",
        "signature"
      ],
      "additionalProperties": false
    },
    "rootReceipt": {
      "type": "object",
      "properties": {
        "kind": {
          "const": "human-admission-receipt"
        },
        "receipt_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "subject_handle": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "root_instance": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "root_key_fingerprint": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "requirement_profile_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "runtime_appraisal_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "context_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "epoch": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "human_auth_time": {
          "type": "string",
          "format": "date-time"
        },
        "not_before": {
          "type": "string",
          "format": "date-time"
        },
        "expires_at": {
          "type": "string",
          "format": "date-time"
        },
        "identity_profile_ref": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "authentication_profile_ref": {
          "type": "string",
```

```
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "signature": {
          "$ref": "#/$defs/signature"
        }
      },
      "required": [
        "kind",
        "receipt_id",
        "subject_handle",
        "root_instance",
        "root_key_fingerprint",
        "policy_digest",
        "requirement_profile_digest",
        "runtime_appraisal_digest",
        "context_digest",
        "epoch",
        "human_auth_time",
        "not_before",
        "expires_at",
        "identity_profile_ref",
        "authentication_profile_ref",
        "signature"
      ],
      "additionalProperties": false
    },
    "runtimeAppraisal": {
      "type": "object",
      "properties": {
        "kind": {
          "const": "runtime-appraisal"
        },
        "appraisal_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "attester_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "instance_key_fingerprint": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "measurement": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "challenge_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "machine_generation": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "issued_at": {
          "type": "string",
          "format": "date-time"
        },
        "expires_at": {
          "type": "string",
          "format": "date-time"
```

```
        },
        "reference_values_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "decision": {
          "enum": [
            "accepted",
            "rejected"
          ]
        },
        "signature": {
          "$ref": "#/$defs/signature"
        }
      },
      "required": [
        "kind",
        "appraisal_id",
        "attester_id",
        "instance_key_fingerprint",
        "measurement",
        "policy_digest",
        "challenge_digest",
        "machine_generation",
        "issued_at",
        "expires_at",
        "reference_values_digest",
        "decision",
        "signature"
      ],
      "additionalProperties": false
    },
    "requirementProfile": {
      "type": "object",
      "properties": {
        "kind": {
          "const": "requirement-profile"
        },
        "scope_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "scope_manifest_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "edition_refs": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "pattern": "^[ -~]+$"
          }
        },
        "source_catalog_digests": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern": "^[0-9a-f]{64}$"
          }
        },
        "selected_ids": {
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "items": {
            "type": "string",
            "minLength": 1,
            "pattern": "^[ -~]+$"
          }
```

```
        },
        "resolved_parameters": {
          "type": "object"
        },
        "evidence_policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "approval_ref": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "profile_status": {
          "enum": [
            "draft",
            "approved"
          ]
        }
      },
      "required": [
        "kind",
        "scope_id",
        "scope_manifest_digest",
        "edition_refs",
        "source_catalog_digests",
        "selected_ids",
        "resolved_parameters",
        "evidence_policy_digest",
        "approval_ref",
        "profile_status"
      ],
      "additionalProperties": false
    },
    "auditEvent": {
      "type": "object",
      "properties": {
        "kind": {
          "const": "authorization-event"
        },
        "event_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "root_instance": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "acting_instance": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "epoch": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "operation_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "decision": {
          "enum": [
            "permit",
            "deny",
            "hold"
```

```
          ]
        },
        "reason_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "event_time": {
          "type": "string",
          "format": "date-time"
        },
        "transaction_id": {
          "type": [
            "string",
            "null"
          ]
        },
        "collector_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "signature": {
          "$ref": "#/$defs/signature"
        }
      },
      "required": [
        "kind",
        "event_id",
        "root_instance",
        "acting_instance",
        "epoch",
        "policy_digest",
        "operation_digest",
        "decision",
        "reason_id",
        "event_time",
        "transaction_id",
        "collector_id",
        "signature"
      ],
      "additionalProperties": false
    },
    "providerEvidence": {
      "type": "object",
      "properties": {
        "kind": {
          "const": "provider-evidence"
        },
        "evidence_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "provider_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "scope_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "subject_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "claim_type": {
          "type": "string",
```

```
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "claim_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "artifact_digests": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern": "^[0-9a-f]{64}$"
          }
        },
        "issued_at": {
          "type": "string",
          "format": "date-time"
        },
        "expires_at": {
          "type": "string",
          "format": "date-time"
        },
        "revocation_ref": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "signature": {
          "$ref": "#/$defs/signature"
        }
      },
      "required": [
        "kind",
        "evidence_id",
        "provider_id",
        "scope_id",
        "subject_id",
        "claim_type",
        "claim_digest",
        "artifact_digests",
        "issued_at",
        "expires_at",
        "revocation_ref",
        "signature"
      ],
      "additionalProperties": false
    }
  },
  "oneOf": [
    {
      "$ref": "#/$defs/rootReceipt"
    },
    {
      "$ref": "#/$defs/runtimeAppraisal"
    },
    {
      "$ref": "#/$defs/requirementProfile"
    },
    {
      "$ref": "#/$defs/auditEvent"
    },
    {
      "$ref": "#/$defs/providerEvidence"
    }
  ]
}
```

C.3. review_objects.schema.json

Source path: schemas/review_objects.schema.json.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "urn:http6:review-objects:v1",
  "title": "HTTP6 review objects: illustrative typed representations",
  "description": "Review profile only. Semantic and cryptographic checks are independent.",
  "$defs": {
    "rule": {
      "type": "object",
      "properties": {
        "operation": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "destination": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "resource": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "data_class": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        }
      },
      "required": [
        "operation",
        "destination",
        "resource",
        "data_class"
      ],
      "additionalProperties": false
    },
    "allowRule": {
      "type": "object",
      "properties": {
        "operation": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$",
          "not": {
            "const": "*"
          }
        },
        "destination": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$",
          "not": {
            "const": "*"
          }
        },
        "resource": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$",
          "not": {
            "const": "*"
          }
        },
        "data_class": {
          "type": "string",
          "minLength": 1,
```

```
          "pattern": "^[ -~]+$",
          "not": {
            "const": "*"
          }
        }
      },
      "required": [
        "operation",
        "destination",
        "resource",
        "data_class"
      ],
      "additionalProperties": false
    },
    "policy": {
      "type": "object",
      "properties": {
        "version": {
          "const": "http6.review-policy.v1"
        },
        "tenant": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "allow": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/allowRule"
          }
        },
        "deny": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/rule"
          }
        },
        "role_snapshot": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "can_mint": {
          "type": "boolean"
        },
        "max_depth": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "templates": {
          "type": "array",
          "items": {
            "type": "string",
            "minLength": 1,
            "pattern": "^[ -~]+$"
          }
        },
        "budget": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        }
      },
      "required": [
        "version",
        "tenant",
        "allow",
        "deny",
        "role_snapshot",
        "can_mint",
        "max_depth",
```

```
        "templates",
        "budget"
      ],
      "additionalProperties": false
    },
    "request": {
      "type": "object",
      "properties": {
        "tenant": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "operation": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "destination": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "resource": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "data_class": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "payload_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "cost": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        }
      },
      "required": [
        "tenant",
        "operation",
        "destination",
        "resource",
        "data_class",
        "payload_digest",
        "cost"
      ],
      "additionalProperties": false
    },
    "grant": {
      "type": "object",
      "properties": {
        "version": {
          "const": "http6.review-grant.v1"
        },
        "instance": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "public_key": {
          "type": "string",
          "pattern": "^04[0-9a-f]{128}$"
        },
        "birth_nonce": {
          "type": "string",
          "pattern": "^[0-9a-f]{32}$"
```

```
        },
        "parent": {
          "anyOf": [
            {
              "type": "string",
              "pattern": "^[0-9a-f]{64}$"
            },
            {
              "type": "null"
            }
          ]
        },
        "root": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "epoch": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "profile": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "issued": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "expires": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "measurement": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "policy": {
          "$ref": "#/$defs/policy"
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        }
      },
      "required": [
        "version",
        "instance",
        "public_key",
        "birth_nonce",
        "parent",
        "root",
        "epoch",
        "profile",
        "issued",
        "expires",
        "measurement",
        "policy",
        "policy_digest"
      ],
      "additionalProperties": false
    },
    "signedGrant": {
      "type": "object",
      "properties": {
        "body": {
          "$ref": "#/$defs/grant"
        },
```

```
        "signature": {
          "type": "string",
          "pattern": "^[0-9a-f]+$",
          "minLength": 2
        }
      },
      "required": [
        "body",
        "signature"
      ],
      "additionalProperties": false
    },
    "action": {
      "type": "object",
      "properties": {
        "instance": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "epoch": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "profile": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "policy_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "lineage_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "channel": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "sequence": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "geometry_digest": {
          "type": "string",
          "pattern": "^[0-9a-f]{64}$"
        },
        "nonce": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "expires": {
          "type": "integer",
          "minimum": 0,
          "maximum": 9007199254740991
        },
        "direction": {
          "const": "request"
        },
        "request": {
          "$ref": "#/$defs/request"
        }
      },
      "required": [
        "instance",
        "epoch",
        "profile",
```

```
        "policy_digest",
        "lineage_digest",
        "channel",
        "sequence",
        "geometry_digest",
        "nonce",
        "expires",
        "direction",
        "request"
      ],
      "additionalProperties": false
    },
    "signedAction": {
      "type": "object",
      "properties": {
        "body": {
          "$ref": "#/$defs/action"
        },
        "signature": {
          "type": "string",
          "pattern": "^[0-9a-f]+$",
          "minLength": 2
        }
      },
      "required": [
        "body",
        "signature"
      ],
      "additionalProperties": false
    },
    "assessmentRecord": {
      "type": "object",
      "properties": {
        "requirement_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "scope_id": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "source_locator": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "implementation_owner": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "implementation_ref": {
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "methods": {
          "type": "array",
          "minItems": 1,
          "uniqueItems": true,
          "items": {
            "enum": [
              "examine",
              "interview",
              "test"
            ]
          }
        },
        "expected": {
```

```
          "type": "string",
          "minLength": 1,
          "pattern": "^[ -~]+$"
        },
        "observed": {
          "type": [
            "string",
            "null"
          ]
        },
        "evidence_digests": {
          "type": "array",
          "items": {
            "type": "string",
            "pattern": "^[0-9a-f]{64}$"
          }
        },
        "status": {
          "enum": [
            "designed",
            "not_assessed",
            "reviewed"
          ]
        },
        "reviewer": {
          "type": [
            "string",
            "null"
          ]
        }
      },
      "required": [
        "requirement_id",
        "scope_id",
        "source_locator",
        "implementation_owner",
        "implementation_ref",
        "methods",
        "expected",
        "observed",
        "evidence_digests",
        "status",
        "reviewer"
      ],
      "additionalProperties": false
    }
  },
  "oneOf": [
    {
      "$ref": "#/$defs/policy"
    },
    {
      "$ref": "#/$defs/request"
    },
    {
      "$ref": "#/$defs/signedGrant"
    },
    {
      "$ref": "#/$defs/signedAction"
    },
    {
      "$ref": "#/$defs/assessmentRecord"
    }
  ]
}
```

Appendix D. Protocol algorithms and state transitions

The following algorithms specify the production integration behavior required by this proposed profile. Uppercase predicates name independently enforced contracts, not implicit success checks. Every external assertion is authenticated and scoped before use. A return of `HOLD` or `DENY` produces no protected effect. The runnable model in Appendix E implements a bounded subset; the Rosario–Wang engine, hardware appraisal, and distributed persistence are explicit adapters. [S1; S2; D2]

D.1. Enroll and prepare a human-originated root

```
ENROLL_HUMAN(account, selected_identity_profile, trusted_device):
  complete the selected proofing, binding, notice, and recovery workflow
  create the protected enrollment relation E without public export of S
  register independently authenticated verifier and device trust anchors
  record enrollment version, lifecycle status, and authorized recovery policy
  assess the complete exposed artifact and transcript privacy profile
  return a non-secret enrollment handle

PREPARE_ROOT(account, proposed_root_key, runtime, policy, requirement_profile):
  verify current organizational approval and selected human prerequisites
  schema-validate policy; resolve its role, operation, and normalizer versions
  resolve the requirement profile and applicable runtime evidence policy
  appraise fresh runtime evidence bound to proposed_root_key and nonce
  construct context c0 naming verifier, channel, root key, code, policy,
      requirement profile, epoch, nonce, lifetime, and approved operation
  authenticate c0 to the trusted Human Admission Client interaction component
  present the consequential approval fields through trusted display
  reconstruct the challenged geometry for this admitted context
  return the challenge and protected context handle
```

The rationale is IA enrollment and authentication, CM admitted configuration, and AC scope approval. A valid person-to-device relationship is established separately from the business scope. The context prevents a genuine ceremony from being applied to an unrelated root or broader policy. Privacy analysis covers what the challenge displays and transmits; the system does not log secret input or private interpretation. [N1; N3; N4]

D.2. Ignite after the private-reference proof

```
IGNITE_ROOT(enrollment_handle, context_handle, witness_package):
  recover the authenticated c0; require unused challenge and current epoch
  verify device/channel binding and the selected authentication factors
  result = RWVerify(E, challenged_geometry, witness_package, c0)
  require result.accepted and all profile-specific human intent checks
  begin authoritative admission transaction
    recheck epoch, policy approvals, runtime appraisal, and nonce status
    consume challenge exactly once
    create root birth record bound to the proposed root public key
    bind policy digest, requirement-profile digest, and evidence-policy digest
    commit root registration, initial lease, and durable admission outcome
  end transaction
  issue an authenticated admission receipt; release only scoped key services
  return root identity and its admitted generation
```

The transaction defines the security linearization point: checks whose truth may change are re-evaluated at the authoritative commit. Retransmission returns the same admitted outcome through an authenticated result-retrieval path. Recovery does not convert a consumed human challenge back to unused. Root renewal and root replacement remain distinct operations, each with its selected human requirements.

D.3. Mint a child directly or through an approved broker

```
MINT_CHILD(parent_request, child_template, target_runtime, requested_policy):
  authenticate the parent request and bind its exact spawning parameters
  validate current ancestry, root receipt, required evidence, and parent lease
  require explicit mint permission, template eligibility, and remaining depth
  validate requested policy and preserve every ancestor ceiling
  reserve global/direct birth counts and any delegated resource allocations
  start the candidate in quarantine; generate its protected instance key
  appraise fresh evidence binding key, code, policy, machine, and birth nonce
  construct child grant with parent-record digest and root-profile digest
  set lifetime no longer than applicable parent and root authorization
  begin authoritative birth transaction
    recheck ancestor revocation, template policy, appraisal, and reservations
    consume one-use birth invitation bound to this candidate key
    register distinct child identity and initial channel generation
    authenticate the grant using the authorized parent's protected mint service
    commit the birth outcome and allocation ownership
  end transaction
  require child proof of possession on its authenticated twin handshake
  activate only the committed instance; preserve quarantine on any failure
```

A broker acts for an identified parent and retains that parent's restrictions. Its signature authority is scoped to the authorized issuance role; it is not a general signing oracle. A copied invitation names one candidate key and is consumed once. A leaf with no mint permission cannot create an authorized descendant even if the same executable contains minting code. A retry with the same birth transaction returns the recorded result. [S1, III.10–III.11]

D.4. Verify a complete ancestry

```
VERIFY_CHAIN(chain, current_registry, selected_profile):
  require bounded length, no duplicate identities, and unambiguous encoding
  verify root admission against an independently trusted admission authority
  require root policy and requirement profile equal the approved selection
  for each parent-to-child edge:
    verify the parent's authenticated grant and authorized issuer role
    verify child binding, parent-record digest, original root, and epoch
    require child template, lifetime, depth, and mint rights within the parent
    verify committed policy bytes and pinned evaluator/role/normalizer meaning
    require current registration, key possession, and applicable runtime evidence
  require current revocation and lease checks for all necessary ancestors
  return the authenticated sequence of ceilings and instance bindings
```

Each link carries authorization rather than a copied human secret. Chain validation establishes provenance; the resource still evaluates the requested action against every ceiling. The reference model uses an out-of-band admission-verifier public key and a trusted registry fixture, which makes those trust inputs inspectable instead of implicitly trusting the root's own signature.

D.5. Authorize and commit the effect

```
AUTHORIZE_EFFECT(received_message, resource_context):
  parse with bounded typed decoding; reject duplicates and ambiguous encodings
  derive actual operation, payload digest, cost, peer, and resource from trusted inputs
  verify complete ancestry and leaf proof of possession
  verify proof binds the exact request, channel, direction, policy, epoch,
      sequence, geometric state, nonce, expiry, and requirement profile
  evaluate every ancestor's complete-request predicate; deny overrides allow
  evaluate current posture and provider facts under pinned interpretation rules
  begin authoritative effect transaction
    recheck epoch/fencing and required mutable authorization facts
    require the issued one-use position and expected channel transition
    atomically reserve all necessary ancestor budgets
    consume the position and create the durable operation/outbox record
    apply a transactional effect or commit an idempotent downstream instruction
    record the outcome and next committed channel state
  end transaction
```

```
  return an authenticated, direction-bound outcome or authorized retry receipt
```

The resource derives the cost and consequential fields rather than accepting a worker's self-report. For an external nontransactional destination, a transactional outbox plus downstream idempotency and reconciliation defines the delivery contract. Consuming a nonce alone does not make an arbitrary remote service exactly-once. An in-flight effect crossing a revocation boundary follows the published commit/fencing semantics; stopping new work and canceling already committed work are separately defined.

D.6. Canonical destination and information-flow decision

```
CHECK_DESTINATION(request, connection):
  normalize scheme, hostname, port, resource, and operation once
  require exact approved endpoint rule and permitted operation/resource pair
  verify the actual authenticated service identity and connected address
  apply all inherited denied identities, domains, ranges, and operations
  verify payload classification and approved transformation evidence
  bind the checked connection to execution; prohibit substitute socket use
  on redirect or route change: reauthorize the new endpoint before disclosure
  return a scoped connection handle, never a general egress capability
```

DNS names, IP ranges, and authenticated service identities answer different questions. The policy records which combinations are required. A hostname that remains on an allowlist does not justify disclosure to a substituted peer. An authorized recipient does not justify sending data outside the approved purpose or classification. [N1, AC-4, SC-7; S1, III.12]

D.7. Revoke, recover, and renew

```
FENCE_SUBTREE(approved_response, target):
  authenticate responder role and exact containment scope
  advance authoritative revocation/fencing state
  stop minting and renewal beneath the target
  notify gates; retain evidence and durable outcomes
  measure the last newly admitted action against the selected cessation bound

REPLACE_CHILD(live_parent, retired_child, approved_template):
  reconcile unfinished business outcomes and preserve consumed positions
  fence the retired generation and reclaim only safe unspent allocations
  perform MINT_CHILD for a fresh key, birth, channel, and appraisal
  resume only the permitted business task

REIGNITE_ROOT(human, approved_profile):
  perform the required new human ceremony and current prerequisites
  establish a new root epoch; retain historical evidence and retired-state fences
  authorize fleet reconstitution under the reviewed replacement plan
```

No heartbeat manufactures a fresh human-authentication time. No backup rolls consumed positions backward. A restored task may retain its business identifier while receiving a new cryptographic identity. Continuity objectives, custodian availability, and key-recovery procedures are specified before the deployment depends on these transitions. [N3, §5; E5; E6]

D.8. Assess a requirement and respond to change

```
ASSESS_REQUIREMENT(source_inventory, assignment, collected_evidence):
  verify approved source selection, exact locator, and resolved parameters
  bind the assignment to actual code, policy, procedure, boundary, and owner
  examine all required objectives and applicable provider dependencies
  authenticate artifact origin and review what the artifacts actually establish
  record observed results and reviewer determinations without filling unknowns
  mark structurally complete only when inventory checks succeed
  mark assessed satisfaction only after the responsible assessment decision

ON_CHANGE(changed_objects):
```

```
    find requirements whose dependency closure intersects changed_objects
    retain historical evidence under its original implementation identity
    open impact reviews and required retests for successor configurations
    apply immediate runtime revocation only as prescribed by approved policy
```

These algorithms connect runtime authority to review evidence without turning every paperwork change into an uncontrolled service interruption. Organizational obligations use actual procedures and accountable reviewers. The official OSCAL model families provide an exchange structure for the resulting system, plan, findings, and corrective-action records; the custom review schema is not presented as an automatically valid OSCAL document. [R2; N2]

Appendix E. Executable reference implementations and complete test source

These listings are the exact review-source files supplied in reference_code/. The authority model uses ECDSA P-256 through the installed cryptography package for synthetic signed records, a trusted test registry, and a local SQLite transaction for modeled effects. It does not implement the Rosario–Wang enrollment engine, hardware key custody, real attestation, live TLS, distributed consensus, or an external exactly-once transaction protocol. Those contracts are specified in A and D. [D2; R4]

The model's role is inspectability: it makes signature checks, fixed policy predicates, ancestry links, expiry, epoch, action binding, and atomic local state transitions executable. register_fixture explicitly creates synthetic appraisal inputs; it is never an acceptable production admission endpoint. A production adapter derives the actual request, peer, cost, channel, and runtime evidence independently of an untrusted application. The stored geometry digest is an input binding, not an implementation of the geometric construction.

The original record-completeness checker and its synthetic tests are retained from the preceding package. Its result concerns record structure and declared state; it does not authenticate evidence or decide NIST compliance. The new catalog importer preserves source structure from an independently supplied local file. The privacy model demonstrates a sufficient small masking construction and exposed-interpretation contrasts; the state explorer checks the stated finite domain.

E.1. authority_model.py

SHA-256: 812757c60fadf86ae8ab29dd93e59866a521f5664a948062808800a0f5b2c685.

```
"""HTTP6 review model, not the production Rosario-Wang engine or a TEE.

Executable scope: signed delegation, restrictive policy intersection, context-bound
requests, and a local SQLite one-use/effect ledger with central ancestry budgets.
Trusted inputs: admission-authority key, registry bindings, current time/epoch,
and independently authenticated actual peer/channel/request. Tests supply fixtures.
TEST-CANON-1 accepts printable ASCII JSON and safe integers; it is not full JCS.
"""
from __future__ import annotations
import copy
import hashlib
import json
import secrets
import sqlite3
from pathlib import Path
from typing import Any
from cryptography.exceptions import InvalidSignature
from cryptography.hazmat.primitives import hashes, serialization
from cryptography.hazmat.primitives.asymmetric import ec

class Rejected(ValueError):
    """The review gate rejects the requested transition."""

def canonical(value: Any) -> bytes:
    def check(x: Any) -> None:
        if x is None or type(x) is bool:
            return
        if type(x) is int and abs(x) <= 9007199254740991:
            return
        if isinstance(x, str) and all(32 <= ord(c) <= 126 for c in x):
            return
        if isinstance(x, list):
            for item in x:
                check(item)
            return
        if isinstance(x, dict) and all(isinstance(k, str) for k in x):
            for k, v in x.items():
```

```
                check(k)
                check(v)
            return
        raise Rejected("TEST-CANON-1 type or range")
    check(value)
    return json.dumps(value, sort_keys=True, ensure_ascii=True,
                      separators=(",", ":"), allow_nan=False).encode("ascii")

def digest(value: Any) -> str:
    return hashlib.sha256(canonical(value)).hexdigest()

def new_key():
    return ec.generate_private_key(ec.SECP256R1())

def public_hex(key) -> str:
    return key.public_key().public_bytes(
        serialization.Encoding.X962,
        serialization.PublicFormat.UncompressedPoint).hex()

def signed(key, domain: str, body: dict) -> dict:
    frozen = copy.deepcopy(body)
    sig = key.sign(canonical([domain, frozen]), ec.ECDSA(hashes.SHA256()))
    return {"body": frozen, "signature": sig.hex()}

def verify(public: str, domain: str, envelope: dict) -> dict:
    try:
        if set(envelope) != {"body", "signature"}:
            raise Rejected("envelope shape")
        key = ec.EllipticCurvePublicKey.from_encoded_point(
            ec.SECP256R1(), bytes.fromhex(public))
        key.verify(bytes.fromhex(envelope["signature"]),
                   canonical([domain, envelope["body"]]),
                   ec.ECDSA(hashes.SHA256()))
        return envelope["body"]
    except (KeyError, TypeError, ValueError, InvalidSignature) as exc:
        raise Rejected("signature or encoding") from exc

def validate_policy(p: dict) -> None:
    required = {"version", "tenant", "allow", "deny", "role_snapshot",
                "can_mint", "max_depth", "templates", "budget"}
    if not isinstance(p, dict) or set(p) != required:
        raise Rejected("policy fields")
    if p["version"] != "http6.review-policy.v1":
        raise Rejected("policy version")
    if type(p["can_mint"]) is not bool:
        raise Rejected("mint flag")
    if not isinstance(p["tenant"], str) or not p["tenant"]:
        raise Rejected("tenant")
    if len(p["role_snapshot"]) != 64:
        raise Rejected("role snapshot")
    if not isinstance(p["templates"], list) or not all(
            isinstance(x, str) and x for x in p["templates"]):
        raise Rejected("templates")
    for name in ("max_depth", "budget"):
        if type(p[name]) is not int or p[name] < 0:
            raise Rejected("nonnegative integer required")
    for effect in ("allow", "deny"):
        if not isinstance(p[effect], list):
            raise Rejected("rules")
        for rule in p[effect]:
            if set(rule) != {"operation", "destination", "resource", "data_class"}:
                raise Rejected("complete request rule required")
            if not all(isinstance(x, str) and x for x in rule.values()):
                raise Rejected("rule values")
            if effect == "allow" and "*" in rule.values():
                raise Rejected("review profile uses exact allow matches")
    canonical(p)

def valid_request(q: dict) -> None:
    fields = {"tenant", "operation", "destination", "resource",
              "data_class", "payload_digest", "cost"}
```

```
    if not isinstance(q, dict) or set(q) != fields:
        raise Rejected("request fields")
    if type(q["cost"]) is not int or q["cost"] < 0:
        raise Rejected("request cost")
    for name in fields - {"cost"}:
        if not isinstance(q[name], str) or not q[name]:
            raise Rejected("request attribute")
    canonical(q)

def policy_allows(p: dict, q: dict) -> bool:
    try:
        validate_policy(p)
        valid_request(q)
        def match(rule):
            return all(v == "*" or q[k] == v for k, v in rule.items())
        return (p["tenant"] == q["tenant"]
                and any(match(r) for r in p["allow"])
                and not any(match(r) for r in p["deny"]))
    except (KeyError, TypeError, ValueError):
        return False

def instance_id(public: str, nonce: str, parent: str | None, epoch: int) -> str:
    return digest(["instance-v1", public, nonce, parent, epoch])

def make_grant(signer, child_key, policy: dict, parent: dict | None,
               profile: str, epoch: int, now: int, expires: int,
               measurement: str) -> dict:
    """Record constructor, NOT a production admission API.

    A root caller is the separately authenticated human-admission authority.
    A child caller must first apply mint_child's authorization checks.
    """
    validate_policy(policy)
    if type(now) is not int or type(expires) is not int or expires <= now:
        raise Rejected("lifetime")
    parent_hash = digest(parent["body"]) if parent else None
    nonce, public = secrets.token_hex(16), public_hex(child_key)
    iid = instance_id(public, nonce, parent_hash, epoch)
    root = parent["body"]["root"] if parent else iid
    body = {"version": "http6.review-grant.v1", "instance": iid,
            "public_key": public, "birth_nonce": nonce, "parent": parent_hash,
            "root": root, "epoch": epoch, "profile": profile,
            "issued": now, "expires": expires, "measurement": measurement,
            "policy": copy.deepcopy(policy), "policy_digest": digest(policy)}
    return signed(signer, "grant-v1", body)

def register_fixture(registry: dict, grant: dict) -> None:
    """TEST HARNESS ONLY: stands for successful external runtime appraisal."""
    b = grant["body"]
    registry[b["instance"]] = {"public_key": b["public_key"],
                               "measurement": b["measurement"],
                               "active": True}

def verify_chain(chain: list[dict], anchor: str, registry: dict,
                 current_epoch: int, now: int, revoked: set[str]) -> list[dict]:
    if not chain or len(chain) > 32:
        raise Rejected("chain length")
    bodies, seen = [], set()
    for pos, envelope in enumerate(chain):
        pub = anchor if pos == 0 else bodies[-1]["public_key"]
        b = verify(pub, "grant-v1", envelope)
        fields = {"version", "instance", "public_key", "birth_nonce", "parent",
                  "root", "epoch", "profile", "issued", "expires",
                  "measurement", "policy", "policy_digest"}
        if set(b) != fields or b["version"] != "http6.review-grant.v1":
            raise Rejected("grant fields")
        validate_policy(b["policy"])
        if b["policy_digest"] != digest(b["policy"]):
            raise Rejected("policy commitment")
        if b["epoch"] != current_epoch or not b["issued"] <= now < b["expires"]:
            raise Rejected("epoch or lifetime")
```

```
        expected_parent = digest(bodies[-1]) if bodies else None
        if b["parent"] != expected_parent:
            raise Rejected("parent link")
        iid = instance_id(b["public_key"], b["birth_nonce"], b["parent"], b["epoch"])
        if b["instance"] != iid or iid in seen or iid in revoked:
            raise Rejected("instance, cycle or revocation")
        if b["root"] != (bodies[0]["root"] if bodies else iid):
            raise Rejected("root identity")
        reg = registry.get(iid, {})
        if (reg.get("active") is not True or reg.get("public_key") != b["public_key"]
                or reg.get("measurement") != b["measurement"]):
            raise Rejected("registered runtime binding")
        if bodies:
            p = bodies[-1]
            if not p["policy"]["can_mint"]:
                raise Rejected("parent may not mint")
            if b["profile"] != p["profile"] or b["expires"] > p["expires"]:
                raise Rejected("profile or lifetime expansion")
            if b["measurement"] not in p["policy"]["templates"]:
                raise Rejected("template")
        for depth, ancestor in enumerate(bodies):
            if pos - depth > ancestor["policy"]["max_depth"]:
                raise Rejected("depth ceiling")
        bodies.append(b)
        seen.add(iid)
    return bodies

def mint_child(chain, anchor, registry, epoch, now, revoked,
               signer, child_key, policy, expires, measurement):
    bodies = verify_chain(chain, anchor, registry, epoch, now, revoked)
    p = bodies[-1]
    if public_hex(signer) != p["public_key"]:
        raise Rejected("wrong issuer key")
    grant = make_grant(signer, child_key, policy, chain[-1], p["profile"],
                       epoch, now, expires, measurement)
    # Staging a binding here is test-only; production uses fresh appraised evidence.
    staged = copy.deepcopy(registry)
    register_fixture(staged, grant)
    verify_chain(chain + [grant], anchor, staged, epoch, now, revoked)
    return grant

class Ledger:
    """Local transactional model. Effects are rows in this same database.

    Every independent connection uses BEGIN IMMEDIATE. No claim of distributed
    consensus or exactly-once external side effects is made by this component.
    """
    def __init__(self, path: str | Path):
        self.path = str(path)
        with self.connect() as db:
            db.executescript('''
              CREATE TABLE IF NOT EXISTS budgets(
                id TEXT PRIMARY KEY, ceiling INTEGER NOT NULL, spent INTEGER NOT NULL);
              CREATE TABLE IF NOT EXISTS challenges(
                token TEXT PRIMARY KEY, binding TEXT NOT NULL, consumed INTEGER NOT NULL);
              CREATE TABLE IF NOT EXISTS effects(
                token TEXT PRIMARY KEY, request_digest TEXT NOT NULL);
            ''')
    def connect(self):
        db = sqlite3.connect(self.path, timeout=20, isolation_level=None)
        db.execute("PRAGMA synchronous=FULL")
        return db
    def add_budget(self, identifier: str, ceiling: int):
        if type(ceiling) is not int or ceiling < 0:
            raise Rejected("budget")
        with self.connect() as db:
            db.execute("INSERT INTO budgets VALUES(?,?,0)", (identifier, ceiling))
    def issue(self, token: str, binding: str):
        with self.connect() as db:
            db.execute("INSERT INTO challenges VALUES(?,?,0)", (token, binding))
    def consume(self, token: str, binding: str, ancestors: list[str],
```

```
                cost: int, request_digest: str) -> None:
        if type(cost) is not int or cost < 0 or not ancestors:
            raise Rejected("cost or ancestry")
        if len(ancestors) != len(set(ancestors)):
            raise Rejected("duplicate ancestry")
        db = self.connect()
        try:
            db.execute("BEGIN IMMEDIATE")
            cur = db.execute(
                "UPDATE challenges SET consumed=1 WHERE token=? AND binding=? AND consumed=0",
                (token, binding))
            if cur.rowcount != 1:
                raise Rejected("unissued, altered or consumed challenge")
            for aid in ancestors:
                cur = db.execute(
                    "UPDATE budgets SET spent=spent+? WHERE id=? AND spent+?<=ceiling",
                    (cost, aid, cost))
                if cur.rowcount != 1:
                    raise Rejected("ancestral budget")
            db.execute("INSERT INTO effects VALUES(?,?)", (token, request_digest))
            db.execute("COMMIT")
        except Exception:
            if db.in_transaction:
                db.execute("ROLLBACK")
            raise
        finally:
            db.close()
    def effect_count(self) -> int:
        with self.connect() as db:
            return db.execute("SELECT COUNT(*) FROM effects").fetchone()[0]

def action_body(leaf: dict, q: dict, channel: str, sequence: int,
                geometry: str, nonce: str, expires: int) -> dict:
    valid_request(q)
    b = leaf["body"]
    return {"instance": b["instance"], "epoch": b["epoch"],
            "profile": b["profile"], "policy_digest": b["policy_digest"],
            "lineage_digest": digest(b), "channel": channel,
            "sequence": sequence, "geometry_digest": geometry,
            "nonce": nonce, "expires": expires, "direction": "request",
            "request": copy.deepcopy(q)}

def authorize(chain, proof, anchor, registry, epoch, now, revoked,
              actual_request, actual_peer, channel, expected_sequence,
              expected_geometry, ledger: Ledger) -> str:
    bodies = verify_chain(chain, anchor, registry, epoch, now, revoked)
    leaf = bodies[-1]
    body = verify(leaf["public_key"], "action-v1", proof)
    expected = action_body(chain[-1], actual_request, channel,
                           expected_sequence, expected_geometry,
                           body["nonce"], body["expires"])
    if body != expected or not now < body["expires"] <= leaf["expires"]:
        raise Rejected("action context")
    if actual_peer != actual_request["destination"]:
        raise Rejected("actual peer")
    if not all(policy_allows(b["policy"], actual_request) for b in bodies):
        raise Rejected("inherited policy")
    # One server-issued nonce for a specific stream position is modeled by token.
    token = digest([epoch, leaf["instance"], channel, expected_sequence])
    ledger.consume(token, digest(body), [b["instance"] for b in bodies],
                   actual_request["cost"], digest(actual_request))
    return "permitted_once_in_local_model"
```

E.2. privacy_model.py

SHA-256: dd741894cf64bd37b86c5162994b06810c034ad068779556aed4d4003db2488a.

```
"""Exhaustive finite illustrations, not a test of a production RW binary.

The fixed circuit C (known to the observer) maps four secrets to two region
indices. Fresh hidden masks make their public distributions identical. Revealing
```

```
those masks with the binary, or publishing the plain indices, distinguishes them.
"""
from collections import Counter
from itertools import product
from fractions import Fraction
import json

def region(secret: int, ceremony: int) -> int:
    return (secret >> (ceremony % 2)) & 1

def distribution(secret: int, rounds: int, reveal_masks: bool = False):
    counts = Counter()
    for masks in product(range(2), repeat=rounds):
        witness = tuple(region(secret, i) ^ masks[i] for i in range(rounds))
        observed = (witness, masks) if reveal_masks else witness
        counts[observed] += 1
    return {k: Fraction(v, 2 ** rounds) for k, v in counts.items()}

def check() -> dict:
    results = {}
    for n in range(1, 7):
        ds = [distribution(s, n) for s in range(4)]
        results[f"fresh_private_mask_rounds_{n}"] = all(d == ds[0] for d in ds)
    public = [distribution(s, 2, True) for s in range(4)]
    results["disclosed_mask_view_distinguishes_secrets"] = len({
        repr(sorted(d.items())) for d in public}) == 4
    paths = [tuple(region(s, i) for i in range(2)) for s in range(4)]
    results["public_deterministic_partition_path_identifies_secret"] = len(set(paths)) == 4
    # A reused mask reveals a relation between two hidden membership indices.
    results["reused_mask_cancels_in_xor"] = all(
        ((a ^ z) ^ (b ^ z)) == (a ^ b)
        for a, b, z in product(range(2), repeat=3))
    return results

if __name__ == "__main__":
    result = check()
    print(json.dumps(result, indent=2))
    raise SystemExit(0 if all(result.values()) else 1)
```

E.3. state_model.py

SHA-256: 1a0f11871eb8c5535fed727cdb724b8a298bcc2991700833f08473efa6e27971.

```
"""Finite abstract transition exploration; cryptographic interfaces are assumed.

State: immutable permission masks for at most three instances, parent identifiers,
root liveness, consumed challenge bits and total consumption. No real RW engine.
"""
from collections import deque
import json

def explore():
    initial = ((3,), (-1,), True, 0, 0)
    todo, seen, transitions = deque([initial]), {initial}, 0
    while todo:
        state = todo.popleft()
        masks, parents, live, used, spent = state
        assert spent <= 2 and spent == used.bit_count()
        assert masks[0] == 3
        assert all((m & ~3) == 0 for m in masks)
        assert all(masks[i] & ~masks[parents[i]] == 0 for i in range(1,len(masks)))
        next_states = []
        if live:
            next_states.append((masks, parents, False, used, spent))
            if len(masks) < 3:
                for p in range(len(masks)):
                    for requested in range(4):
                        child = masks[p] & requested
                        next_states.append((masks+(child,),parents+(p,),live,used,spent))
            for p, permissions in enumerate(masks):
                for op in (1,2):
```

```
                    for nonce in (0,1):
                        if permissions & op and not (used & (1 << nonce)):
                            next_states.append((masks,parents,live,used|(1<<nonce),spent+1))
        # Fenced states have no effect-producing transition.
        for nxt in next_states:
            transitions += 1
            if nxt not in seen:
                seen.add(nxt);todo.append(nxt)
    return {'model':'finite-attenuation-and-consumption-v1','states':len(seen),
            'transitions':transitions,'invariants':'all checked in this finite model',
            'bounds':{'max_instances':3,'permission_bits':2,'challenge_positions':2},
            'production_assessment':False}
if __name__=='__main__':
    print(json.dumps(explore(),indent=2))
```

E.4. import_oscal_catalog.py

SHA-256: 65019610eb8aff835975bb8ea64b918526ab0d2d570087a01ad36c9ec770d279.

```
#!/usr/bin/env python3
"""Extract source-addressable OSCAL catalog entries from a local pinned file.

This is an inventory aid, not an automatic control selection or assessment.
Original statements/parameters are retained. A catalog without assessment
objectives still needs its separate assessment-procedure source and approval.
"""
from __future__ import annotations
import argparse
import hashlib
import json
from pathlib import Path

MAX_BYTES = 100_000_000

def extract(path: Path, expected_sha256: str, expected_version: str) -> dict:
    raw = path.read_bytes()
    if len(raw) > MAX_BYTES:
        raise ValueError('catalog exceeds configured size limit')
    observed = hashlib.sha256(raw).hexdigest()
    if observed != expected_sha256.lower():
        raise ValueError('catalog digest mismatch')
    document = json.loads(raw)
    catalog = document['catalog']
    metadata = catalog['metadata']
    if metadata['version'] != expected_version:
        raise ValueError('catalog edition mismatch')
    entries, seen = [], set()
    def controls(items, groups):
        for control in items:
            cid = control['id']
            if cid in seen:
                raise ValueError('duplicate control identifier: ' + cid)
            seen.add(cid)
            props = control.get('props', [])
            states = [p['value'] for p in props if p.get('name') == 'status']
            entries.append({
                'source_control_id': cid, 'title': control.get('title'),
                'group_path': list(groups), 'source_status': states or ['unspecified'],
                'parameters': control.get('params', []),
                'parts': control.get('parts', []), 'properties': props,
                'links': control.get('links', []),
                'applicability': 'pending_approved_selection',
                'assessment_objective_source': None,
                'implementation_assignment': None,
                'observed_result': None})
            controls(control.get('controls', []), groups)
    def groups(items, ancestry):
        for group in items:
            here = ancestry + [group['id']]
            controls(group.get('controls', []), here)
            groups(group.get('groups', []), here)
```

```
    controls(catalog.get('controls', []), [])
    groups(catalog.get('groups', []), [])
    return {'record_kind': 'unselected_source_catalog_extract',
            'source_file': path.name, 'source_digest': observed,
            'source_version': metadata['version'],
            'source_title': metadata.get('title'),
            'oscal_version': metadata.get('oscal-version'),
            'approved_by': None, 'scope_id': None,
            'source_control_count': len(entries), 'entries': entries,
            'next_steps': ['approve selection and boundary',
                           'resolve parameters',
                           'import applicable assessment objectives',
                           'assign implementations and actual evidence']}

def main():
    p = argparse.ArgumentParser(description=__doc__)
    p.add_argument('catalog', type=Path)
    p.add_argument('--sha256', required=True)
    p.add_argument('--version', required=True)
    p.add_argument('--output', required=True, type=Path)
    args = p.parse_args()
    try:
        result = extract(args.catalog, args.sha256, args.version)
        args.output.write_text(json.dumps(result, indent=2)+'\n',encoding='utf-8')
    except (ValueError, KeyError, TypeError, OSError) as exc:
        p.exit(2, 'Import rejected: '+str(exc)+'\n')
if __name__=='__main__':
    main()
```

E.5. check_requirement_closure.py

SHA-256: 0da8cae70fec9e1cce959783566f00d279dc8a20f7fde69420af70e834451b4f.

```
#!/usr/bin/env python3
"""Check completeness of an externally selected requirement inventory and evidence register.

This checks record structure and declared state. It does not authenticate artifacts,
verify NIST applicability, establish evidence truth, or perform a compliance assessment.
The caller must provide an independently approved, complete selected inventory.

Usage:
  python check_requirement_closure.py inventory.json register.json --as-of 2026-11-09T00:00:00Z
Exit codes: 0 structurally ready; 1 not ready; 2 invalid input.
"""
from __future__ import annotations
import argparse
import datetime as dt
import json
import re
import sys
from pathlib import Path
from typing import Any

HEX64 = re.compile(r"^[0-9a-fA-F]{64}$")
METHODS = {"examine", "interview", "test"}
UNRESOLVED = {"", "tbd", "pending", "unresolved", "unknown", "not_assessed"}

def populated(value: Any) -> bool:
    if value is None:
        return False
    if isinstance(value, str):
        return value.strip().lower() not in UNRESOLVED
    if isinstance(value, (list, dict)):
        return bool(value) and all(populated(v) for v in (value.values() if isinstance(value, dict) else
        ↪  value))
    return True  # False and zero may be explicit valid parameter values.

def time_value(value: Any) -> dt.datetime:
    if not isinstance(value, str):
        raise ValueError("timestamp must be an ISO-8601 string")
    result = dt.datetime.fromisoformat(value.replace("Z", "+00:00"))
```

```
    if result.tzinfo is None:
        raise ValueError("timestamp must include timezone")
    return result.astimezone(dt.timezone.utc)

def unique_named(items: Any, key: str, path: str, errors: list[str]) -> dict[str, dict[str, Any]]:
    if not isinstance(items, list):
        errors.append(f"{path}: expected list")
        return {}
    indexed = {}
    for i, item in enumerate(items):
        if not isinstance(item, dict) or not isinstance(item.get(key), str) or not item[key].strip():
            errors.append(f"{path}[{i}]: missing string {key}")
            continue
        identifier = item[key]
        if identifier in indexed:
            errors.append(f"{path}: duplicate {identifier}")
        indexed[identifier] = item
    return indexed

def check(inventory: dict[str, Any], register: dict[str, Any], as_of: dt.datetime) -> dict[str, Any]:
    errors: list[str] = []
    if not isinstance(inventory, dict) or not isinstance(register, dict):
        raise ValueError("inventory and register must be JSON objects")
    if as_of.tzinfo is None:
        raise ValueError("as_of requires a timezone")
    for field in ("scope_id", "source_version", "approved_by"):
        if not populated(inventory.get(field)):
            errors.append(f"inventory.{field}: approved value required")
    digest = inventory.get("source_digest", "")
    if not isinstance(digest, str) or not HEX64.fullmatch(digest):
        errors.append("inventory.source_digest: 64-hex source digest required")
    if register.get("scope_id") != inventory.get("scope_id"):
        errors.append("register.scope_id: inventory mismatch")
    if register.get("inventory_digest") != digest:
        errors.append("register.inventory_digest: inventory mismatch")
    reqs = unique_named(inventory.get("requirements"), "id", "inventory.requirements", errors)
    records = unique_named(register.get("records"), "requirement_id", "register.records", errors)
    if not reqs:
        errors.append("inventory.requirements: nonempty approved selection required")
    missing = sorted(set(reqs) - set(records))
    extra = sorted(set(records) - set(reqs))
    if missing:
        errors.append("register: missing requirements " + ", ".join(missing))
    if extra:
        errors.append("register: unselected requirements " + ", ".join(extra))
    own_ok: dict[str, bool] = {}
    dependencies: dict[str, list[str]] = {}
    for rid, req in reqs.items():
        if rid not in records:
            own_ok[rid] = False
            dependencies[rid] = []
            continue
        rec = records[rid]
        before = len(errors)
        if rec.get("scope_id") != inventory.get("scope_id"):
            errors.append(f"{rid}: wrong evidence boundary")
        for field in ("implementation_owner", "implementation_ref"):
            if not populated(rec.get(field)):
                errors.append(f"{rid}.{field}: required")
        if rec.get("assessment_status") != "satisfied":
            errors.append(f"{rid}: assessment status must be satisfied; risk acceptance alone is not
            ↪  satisfaction")
        parameters = rec.get("parameters", {})
        if not isinstance(parameters, dict):
            errors.append(f"{rid}.parameters: expected object")
            parameters = {}
        required_params = req.get("parameters", [])
        if not isinstance(required_params, list) or not all(isinstance(x,str) for x in required_params):
            errors.append(f"{rid}: parameter inventory must be a string list")
            required_params = []
        for name in required_params:
```

```
            if name not in parameters or not populated(parameters[name]):
                errors.append(f"{rid}.parameters.{name}: unresolved")
        evidence = unique_named(rec.get("evidence", []), "id", f"{rid}.evidence", errors)
        if not evidence:
            errors.append(f"{rid}: evidence required")
        for eid, ev in evidence.items():
            if not isinstance(ev.get("digest"), str) or not HEX64.fullmatch(ev["digest"]):
                errors.append(f"{rid}.{eid}: artifact digest required")
            if ev.get("scope_id") != inventory.get("scope_id"):
                errors.append(f"{rid}.{eid}: evidence scope mismatch")
            if ev.get("implementation_ref") != rec.get("implementation_ref"):
                errors.append(f"{rid}.{eid}: evidence implementation mismatch")
            if not populated(ev.get("collector")):
                errors.append(f"{rid}.{eid}: collector required")
            try:
                collected = time_value(ev.get("collected_at"))
                valid_until = time_value(ev.get("valid_until"))
                if collected > as_of or valid_until <= as_of or valid_until <= collected:
                    errors.append(f"{rid}.{eid}: stale, future, or invalid evidence interval")
            except (ValueError, TypeError) as exc:
                errors.append(f"{rid}.{eid}: invalid evidence time ({exc})")
        objs = unique_named(rec.get("objectives", []), "id", f"{rid}.objectives", errors)
        expected_obj = req.get("objectives", [])
        if not isinstance(expected_obj,list) or not expected_obj or not all(isinstance(x,str) for x in
        ↪  expected_obj):
            errors.append(f"{rid}: nonempty objective inventory required")
            expected_obj = []
        if set(objs) != set(expected_obj):
            errors.append(f"{rid}: objective set mismatch")
        for oid, obj in objs.items():
            if obj.get("determination") != "satisfied":
                errors.append(f"{rid}.{oid}: objective not satisfied")
            if not populated(obj.get("reviewer")) or not populated(obj.get("observed")):
                errors.append(f"{rid}.{oid}: actual observation and reviewer required")
            methods = obj.get("methods", [])
            if not isinstance(methods, list) or not methods or not all(isinstance(m,str) and m in METHODS
            ↪  for m in methods):
                errors.append(f"{rid}.{oid}: valid justified assessment method required")
            refs = obj.get("evidence_ids", [])
            if not isinstance(refs,list) or not refs or not all(isinstance(x,str) and x in evidence for x
            ↪  in refs):
                errors.append(f"{rid}.{oid}: current evidence reference required")
        deps = rec.get("dependencies", [])
        if not isinstance(deps,list) or not all(isinstance(x,str) for x in deps):
            errors.append(f"{rid}.dependencies: expected requirement identifier list")
            deps = []
        dependencies[rid] = deps
        for dep in deps:
            if dep not in reqs:
                errors.append(f"{rid}: unresolved dependency {dep}")
        own_ok[rid] = len(errors) == before
    closed: dict[str,bool] = {}
    active: set[str] = set()
    def resolve(rid: str) -> bool:
        if rid in closed:
            return closed[rid]
        if rid in active:
            errors.append(f"{rid}: circular dependency needs explicit resolution")
            return False
        if rid not in own_ok:
            return False
        active.add(rid)
        # Evaluate all dependencies to expose all cycles even when local evidence fails.
        dep_ok = [resolve(dep) for dep in dependencies.get(rid, [])]
        result = own_ok[rid] and all(dep_ok)
        active.remove(rid)
        closed[rid] = result
        return result
    for rid in reqs:
        resolve(rid)
    ready = bool(reqs) and not errors and all(closed.values())
```

```
    return {
        "checker_scope": "record_completeness_only_not_a_NIST_or_product_assessment",
        "as_of": as_of.isoformat(),
        "structurally_ready": ready,
        "requirements_selected": len(reqs),
        "records_present_for_selection": len(set(reqs)&set(records)),
        "records_structurally_closed": sum(closed.values()),
        "missing_requirement_ids": missing,
        "errors": errors,
        "per_requirement_structural_closure": closed,
    }

def main() -> int:
    parser = argparse.ArgumentParser(description=__doc__)
    parser.add_argument("inventory", type=Path)
    parser.add_argument("register", type=Path)
    parser.add_argument("--as-of", required=True, help="Timezone-aware ISO-8601 assessment time")
    args=parser.parse_args()
    try:
        result=check(json.loads(args.inventory.read_text()),json.loads(args.register.read_text()),time_v⌋
        ↪  alue(args.as_of))
    except (OSError, ValueError, TypeError, KeyError) as exc:
        print(json.dumps({"input_error": str(exc)},indent=2),file=sys.stderr)
        return 2
    print(json.dumps(result,indent=2))
    return 0 if result["structurally_ready"] else 1
if __name__ == "__main__":
    raise SystemExit(main())
```

E.6. run_public_example.py

SHA-256: 40186c6828a1b57d677772f139f767e34ff26bf667e73f7770f67ff8cf28704f.

```
"""Replay the public synthetic records; no private signing keys are needed."""
import json
from pathlib import Path
from tempfile import TemporaryDirectory
from authority_model import Ledger, digest, authorize, Rejected

ROOT = Path(__file__).resolve().parents[1]

def run():
    read = lambda name: json.loads((ROOT/'examples'/name).read_text())
    chain = [read('review_root_grant.json'),read('review_child_grant.json')]
    proof = read('review_action_proof.json')
    fixture = read('review_public_registry.json')
    request = read('review_request.json')
    with TemporaryDirectory() as directory:
        ledger = Ledger(Path(directory)/'review.sqlite')
        for g in chain:
            ledger.add_budget(g['body']['instance'],g['body']['policy']['budget'])
        b=proof['body']
        token=digest([b['epoch'],b['instance'],b['channel'],b['sequence']])
        ledger.issue(token,digest(b))
        args=dict(chain=chain,proof=proof,
                  anchor=fixture['admission_verifier_public_key'],
                  registry=fixture['registry'],epoch=fixture['epoch'],
                  now=fixture['now'],revoked=set(fixture['revoked']),
                  actual_request=request,actual_peer=request['destination'],
                  channel=b['channel'],expected_sequence=b['sequence'],
                  expected_geometry=b['geometry_digest'],ledger=ledger)
        first=authorize(**args)
        try:
            authorize(**args)
            replay='unexpected_accept'
        except Rejected:
            replay='rejected'
        return {'fixture_kind':fixture['fixture_kind'],'first':first,
                'replay':replay,'local_effect_count':ledger.effect_count()}

if __name__=='__main__':
```

```
    result=run(); print(json.dumps(result,indent=2))
    raise SystemExit(0 if result['replay']=='rejected'
                     and result['local_effect_count']==1 else 1)
```

E.7. test_authority_model.py

SHA-256: 6a451c3018dabff8d0288b4301dd50493f6f965720c6204012b219b4c3cde9f5.

```
"""Synthetic review-model tests. No production authentication or attestation."""
import copy
from concurrent.futures import ThreadPoolExecutor
import pytest
from authority_model import (
    Rejected, canonical, digest, new_key, public_hex, signed, make_grant,
    register_fixture, verify_chain, mint_child, policy_allows, Ledger,
    action_body, authorize, instance_id)
from privacy_model import check

RULE = {"operation": "report.publish", "destination": "reports.example",
        "resource": "daily", "data_class": "aggregate"}

def policy(mint=True, depth=4, budget=10):
    return {"version": "http6.review-policy.v1", "tenant": "acme",
            "allow": [dict(RULE)], "deny": [], "role_snapshot": "a" * 64,
            "can_mint": mint, "max_depth": depth,
            "templates": ["worker-v1"], "budget": budget}

def request():
    return {"tenant": "acme", **RULE, "payload_digest": "b" * 64, "cost": 1}

@pytest.fixture
def family(tmp_path):
    anchor, rootkey, childkey = new_key(), new_key(), new_key()
    root = make_grant(anchor, rootkey, policy(), None, "c"*64, 42, 10, 100, "root-v1")
    registry = {}
    register_fixture(registry, root)
    child = mint_child([root], public_hex(anchor), registry, 42, 20, set(),
                       rootkey, childkey, policy(False, 0), 90, "worker-v1")
    register_fixture(registry, child)
    ledger = Ledger(tmp_path / 'state.sqlite')
    for g in (root, child):
        ledger.add_budget(g['body']['instance'], g['body']['policy']['budget'])
    return dict(anchor=anchor, rootkey=rootkey, childkey=childkey,
                chain=[root, child], registry=registry, ledger=ledger)

def verify_family(f, **kwargs):
    args = dict(chain=f['chain'], anchor=public_hex(f['anchor']),
                registry=f['registry'], current_epoch=42, now=25, revoked=set())
    args.update(kwargs)
    return verify_chain(**args)

def prepare_action(f, q=None, seq=0):
    q = q or request()
    leaf = f['chain'][-1]
    b = action_body(leaf, q, 'channel-1', seq, 'd'*64, 'nonce-1', 70)
    proof = signed(f['childkey'], 'action-v1', b)
    token = digest([42, leaf['body']['instance'], 'channel-1', seq])
    f['ledger'].issue(token, digest(b))
    return proof

def execute(f, proof, **kwargs):
    args = dict(chain=f['chain'], proof=proof, anchor=public_hex(f['anchor']),
                registry=f['registry'], epoch=42, now=25, revoked=set(),
                actual_request=request(), actual_peer='reports.example',
                channel='channel-1', expected_sequence=0,
                expected_geometry='d'*64, ledger=f['ledger'])
    args.update(kwargs)
    return authorize(**args)

def test_chain_and_authorized_effect(family):
    assert len(verify_family(family)) == 2
```

```
    assert execute(family, prepare_action(family)).startswith('permitted_once')
    assert family['ledger'].effect_count() == 1

def test_replay(family):
    p = prepare_action(family)
    execute(family, p)
    with pytest.raises(Rejected): execute(family, p)
    assert family['ledger'].effect_count() == 1

@pytest.mark.parametrize('field,value', [
    ('operation','ledger.export'), ('destination','external.example'),
    ('resource','raw-ledger'), ('data_class','raw'), ('tenant','other')])
def test_complete_request_restrictions(field, value):
    q = request(); q[field] = value
    assert not policy_allows(policy(), q)

def test_deny_overrides():
    p = policy(); p['deny'] = [dict(RULE)]
    assert not policy_allows(p, request())

def test_missing_attribute():
    q=request(); del q['resource']
    assert not policy_allows(policy(), q)

def test_unknown_policy_field():
    p=policy(); p['ignore_ancestors']=True
    assert not policy_allows(p,request())

def test_child_local_expansion_cannot_override_root(family):
    f=family
    extra={**RULE,'operation':'ledger.export'}
    p=policy(False,0);p['allow'].append(extra)
    root=f['chain'][0]
    child=make_grant(f['rootkey'],f['childkey'],p,root,'c'*64,42,20,90,'worker-v1')
    register_fixture(f['registry'],child)
    f['chain']=[root,child]
    f['ledger'].add_budget(child['body']['instance'],10)
    q=request();q['operation']='ledger.export'
    proof=prepare_action(f,q)
    with pytest.raises(Rejected):execute(f,proof,actual_request=q)
    assert f['ledger'].effect_count()==0

@pytest.mark.parametrize('fault', [
    'policy', 'parent', 'profile', 'signature', 'root', 'expired',
    'revoked', 'epoch', 'registry', 'anchor', 'omitted_root'])
def test_reject_bad_chains(family, fault):
    f=family; chain=copy.deepcopy(f['chain']); kwargs={}
    if fault=='policy':chain[-1]['body']['policy']['allow']=[]
    elif fault=='parent':chain[-1]['body']['parent']='0'*64
    elif fault=='profile':
        b=chain[-1]['body'];b['profile']='e'*64
        chain[-1]=signed(f['rootkey'],'grant-v1',b)
    elif fault=='signature':chain[-1]['signature']='00'
    elif fault=='root':chain[-1]['body']['root']='0'*64
    elif fault=='expired':kwargs['now']=100
    elif fault=='revoked':kwargs['revoked']={chain[0]['body']['instance']}
    elif fault=='epoch':kwargs['current_epoch']=43
    elif fault=='registry':f['registry'][chain[-1]['body']['instance']]['active']=False
    elif fault=='anchor':kwargs['anchor']=public_hex(new_key())
    elif fault=='omitted_root':chain=chain[1:]
    with pytest.raises(Rejected):verify_family(f,chain=chain,**kwargs)

@pytest.mark.parametrize('fault', ['leaf_mints','wrong_template','long_life','wrong_signer'])
def test_reject_unauthorized_minting(family,fault):
    f=family;chain=f['chain'][:1]; signer=f['rootkey'];expiry=90;measurement='worker-v1'
    if fault=='leaf_mints':chain=f['chain'];signer=f['childkey']
    if fault=='wrong_template':measurement='unapproved'
    if fault=='long_life':expiry=110
    if fault=='wrong_signer':signer=new_key()
    with pytest.raises(Rejected):
        mint_child(chain,public_hex(f['anchor']),f['registry'],42,25,set(),
```

```
                    signer,new_key(),policy(False,0),expiry,measurement)

@pytest.mark.parametrize('field,value', [
    ('actual_peer','external.example'), ('channel','other-channel'),
    ('expected_sequence',1), ('expected_geometry','e'*64), ('epoch',43)])
def test_action_binding(family,field,value):
    with pytest.raises(Rejected):execute(family,prepare_action(family),**{field:value})

def test_action_payload_binding(family):
    q=request();q['payload_digest']='e'*64
    with pytest.raises(Rejected):execute(family,prepare_action(family),actual_request=q)

def test_unissued_challenge(family):
    b=action_body(family['chain'][-1],request(),'channel-1',0,'d'*64,'nonce-1',70)
    with pytest.raises(Rejected):execute(family,signed(family['childkey'],'action-v1',b))

def test_budget_rollback(family):
    f=family
    with f['ledger'].connect() as db:
        db.execute('UPDATE budgets SET ceiling=0 WHERE id=?',
                   (f['chain'][0]['body']['instance'],))
    proof=prepare_action(f)
    with pytest.raises(Rejected):execute(f,proof)
    with f['ledger'].connect() as db:
        assert db.execute('SELECT SUM(spent) FROM budgets').fetchone()[0]==0
        assert db.execute('SELECT consumed FROM challenges').fetchone()[0]==0
    assert f['ledger'].effect_count()==0

def test_concurrent_replay_one_effect(tmp_path):
    path=tmp_path/'race.sqlite';ledger=Ledger(path)
    ledger.add_budget('root',100);ledger.issue('token','binding')
    def run(_):
        try:
            Ledger(path).consume('token','binding',['root'],1,'request')
            return True
        except Rejected:return False
    with ThreadPoolExecutor(max_workers=8) as pool:results=list(pool.map(run,range(32)))
    assert sum(results)==1
    assert ledger.effect_count()==1

def test_shared_budget_race(tmp_path):
    path=tmp_path/'budget.sqlite';ledger=Ledger(path);ledger.add_budget('root',1)
    for i in range(8):ledger.issue(str(i),'binding')
    def run(i):
        try:
            Ledger(path).consume(str(i),'binding',['root'],1,'request');return True
        except Rejected:return False
    with ThreadPoolExecutor(max_workers=8) as pool:results=list(pool.map(run,range(8)))
    assert sum(results)==1

def test_restart_does_not_reset_consumption(tmp_path):
    path=tmp_path/'restart.sqlite';a=Ledger(path);a.add_budget('root',2);a.issue('t','b')
    a.consume('t','b',['root'],1,'q')
    with pytest.raises(Rejected):Ledger(path).consume('t','b',['root'],1,'q')

def test_canonical_order():
    assert canonical({'b':2,'a':1})==b'{"a":1,"b":2}'

@pytest.mark.parametrize('value',[1.2,float('nan'),'non-ascii-\u00e9',2**60])
def test_restricted_encoding(value):
    with pytest.raises(Rejected):canonical(value)

def test_finite_privacy_cases():
    assert all(check().values())
```

E.8. test_catalog_import.py

SHA-256: 331d0a2d71fab4785735d909d56b03624dc4cd4d6229f17a035a46f4f4ec0e0d.

```
import hashlib
import json
import pytest
from import_oscal_catalog import extract

@pytest.fixture
def catalog(tmp_path):
    doc={'catalog':{'metadata':{'version':'test-v1','title':'Synthetic','oscal-version':'1.1.2'},
       'groups':[{'id':'ac','controls':[{'id':'ac-3','title':'Access',
       'params':[{'id':'ac-3_prm_1','label':'source value'}],
       'parts':[{'id':'ac-3_smt','name':'statement','prose':'synthetic statement'}],
       'controls':[{'id':'ac-3.1','title':'Synthetic enhancement'}]}]}]}}
    path=tmp_path/'catalog.json';path.write_text(json.dumps(doc))
    return path,hashlib.sha256(path.read_bytes()).hexdigest()

def test_import_preserves_source(catalog):
    path,h=catalog;r=extract(path,h,'test-v1')
    assert r['source_control_count']==2
    assert r['entries'][0]['parts'][0]['id']=='ac-3_smt'
    assert r['approved_by'] is None
    assert r['entries'][0]['assessment_objective_source'] is None

def test_digest_rejection(catalog):
    with pytest.raises(ValueError):extract(catalog[0],'0'*64,'test-v1')

def test_version_rejection(catalog):
    with pytest.raises(ValueError):extract(*catalog,'wrong-version')

def test_duplicate_rejection(catalog):
    path,_=catalog;doc=json.loads(path.read_text())
    doc['catalog']['groups'][0]['controls'].append({'id':'ac-3','title':'Duplicate'})
    path.write_text(json.dumps(doc));h=hashlib.sha256(path.read_bytes()).hexdigest()
    with pytest.raises(ValueError):extract(path,h,'test-v1')
```

E.9. test_closure_checker.py

SHA-256: 3cfd43183f602d46448a75b77ec6cc4db958978b3ac19dba24203708eefce43f.

```
"""Synthetic checker tests. These are not HTTP6 implementation or NIST assessment tests."""
import copy,json,hashlib
from pathlib import Path
from check_requirement_closure import check,time_value
ROOT=Path(__file__).resolve().parent
AS_OF=time_value('2026-11-09T00:00:00Z')
inv={"scope_id":"synthetic-fixture-only","source_version":"synthetic-v1-not-a-NIST-catalog","approved_by
↪  ":"synthetic-reviewer","source_digest":hashlib.sha256(b'synthetic fixture not authoritative
↪  NIST').hexdigest(),"requirements":[{"id":"SYN-001","parameters":["lease_seconds"],"objectives":["SYN-
↪  001-OBJ"]},{"id":"SYN-002","parameters":[],"objectives":["SYN-002-OBJ"]}]}
records=[]
for rid in ['SYN-001','SYN-002']:
    records.append({"requirement_id":rid,"scope_id":inv['scope_id'],"implementation_owner":"synthetic-ow
    ↪  ner","implementation_ref":"fixture-engine/v1","assessment_status":"satisfied","parameters":{"lea
    ↪  se_seconds":60} if rid=='SYN-001' else {},"dependencies":["SYN-001"] if rid=='SYN-002' else
    ↪  [],"objectives":[{"id":rid+'-OBJ',"determination":"satisfied","reviewer":"synthetic-reviewer","o
    ↪  bserved":"Synthetic record field for parser tests
    ↪  only","methods":["test"],"evidence_ids":[rid+'-EV']}],"evidence":[{"id":rid+'-EV',"digest":hashl
    ↪  ib.sha256(rid.encode()).hexdigest(),"scope_id":inv['scope_id'],"implementation_ref":"fixture-eng
    ↪  ine/v1","collector":"synthetic-collector","collected_at":"2026-11-01T00:00:00Z","valid_until":"2
    ↪  026-12-01T00:00:00Z"}]})
reg={"scope_id":inv['scope_id'],"inventory_digest":inv['source_digest'],"records":records}
(ROOT/'synthetic_inventory.json').write_text(json.dumps(inv,indent=2))
(ROOT/'synthetic_register.json').write_text(json.dumps(reg,indent=2))
tests=[]
def test(name,mutate,expected=False):
    i=copy.deepcopy(inv);r=copy.deepcopy(reg);mutate(i,r)
    out=check(i,r,AS_OF)
```

```
    assert out['structurally_ready'] is expected,(name,out)
    tests.append({'test':name,'expected_ready':expected,'actual_ready':out['structurally_ready'],'passed
    ↪  ':True,'errors_detected':out['errors']})
test('complete_synthetic_records',lambda i,r:None,True)
test('missing_requirement',lambda i,r:r['records'].pop())
test('duplicate_record',lambda i,r:r['records'].append(copy.deepcopy(r['records'][0])))
test('unresolved_parameter',lambda i,r:r['records'][0]['parameters'].update(lease_seconds=None))
test('missing_objective',lambda i,r:r['records'][0].update(objectives=[]))
test('failed_objective',lambda
↪  i,r:r['records'][0]['objectives'][0].update(determination='not_satisfied'))
test('stale_evidence',lambda
↪  i,r:r['records'][0]['evidence'][0].update(valid_until='2026-11-08T00:00:00Z'))
test('wrong_environment',lambda
↪  i,r:r['records'][0]['evidence'][0].update(implementation_ref='different-engine/v2'))
test('missing_dependency',lambda i,r:r['records'][0].update(dependencies=['SYN-404']))
test('circular_dependency',lambda i,r:r['records'][0].update(dependencies=['SYN-002']))
test('unapproved_inventory',lambda i,r:i.update(approved_by=None))
test('empty_inventory',lambda i,r:i.update(requirements=[]))
test('risk_acceptance_not_satisfaction',lambda
↪  i,r:r['records'][0].update(assessment_status='risk_accepted'))
test('missing_observation',lambda i,r:r['records'][0]['objectives'][0].update(observed=None))
test('wrong_scope',lambda i,r:r.update(scope_id='another-boundary'))
test('invalid_source_digest',lambda i,r:i.update(source_digest='not-a-hash'))
test('future_evidence',lambda
↪  i,r:r['records'][0]['evidence'][0].update(collected_at='2026-11-10T00:00:00Z'))
test('zero_is_resolved_parameter',lambda i,r:r['records'][0]['parameters'].update(lease_seconds=0),True)
report={'scope':'synthetic_record_checker_tests_only','count':len(tests),'passed':len(tests),'tests':tes
↪  ts}
(ROOT/'closure_checker_test_results.json').write_text(json.dumps(report,indent=2))
print(f'{len(tests)} synthetic checker tests passed; no HTTP6 implementation assessment performed.')
```

E.10. regenerate_public_example.py

SHA-256: 790b69e308f9aa3decc36288ef6190d82bebe3d1a4ead463873c54f5184b474e.

```
"""Rebuild the public synthetic signed records with fresh ephemeral keys.

The script reads the request, policies, epoch, times, measurements, profile,
channel, sequence, geometry digest, and action nonce from the current records in
`examples/`, generates new P-256 keys, and rewrites review_root_grant.json,
review_child_grant.json, review_action_proof.json, and review_public_registry.json.
Private keys exist only in memory and are discarded when the script exits.
Birth nonces are drawn fresh by make_grant. The output is then replayed through
run_public_example.run(), and the script exits 1 unless the first use is accepted
and the replay is rejected.
"""
import copy
import json
from pathlib import Path
from authority_model import (new_key, public_hex, signed, make_grant, mint_child,
                             register_fixture, action_body)
from run_public_example import run

EXAMPLES = Path(__file__).resolve().parents[1] / 'examples'

def read(name):
    return json.loads((EXAMPLES / name).read_text())

def write(name, value):
    (EXAMPLES / name).write_text(json.dumps(value, indent=2) + '\n')

def regenerate():
    old_root = read('review_root_grant.json')['body']
    old_child = read('review_child_grant.json')['body']
    old_proof = read('review_action_proof.json')['body']
    old_fixture = read('review_public_registry.json')
    root_policy = read('review_policy.json')
    request = read('review_request.json')
    child_policy = copy.deepcopy(old_child['policy'])
    epoch = old_fixture['epoch']
```

```
    anchor, root_key, child_key = new_key(), new_key(), new_key()
    root = make_grant(anchor, root_key, root_policy, None, old_root['profile'],
                      epoch, old_root['issued'], old_root['expires'],
                      old_root['measurement'])
    registry = {}
    register_fixture(registry, root)
    child = mint_child([root], public_hex(anchor), registry, epoch,
                       old_child['issued'], set(old_fixture['revoked']),
                       root_key, child_key, child_policy, old_child['expires'],
                       old_child['measurement'])
    register_fixture(registry, child)
    proof = signed(child_key, 'action-v1',
                   action_body(child, request, old_proof['channel'],
                               old_proof['sequence'], old_proof['geometry_digest'],
                               old_proof['nonce'], old_proof['expires']))
    fixture = {'fixture_kind': old_fixture['fixture_kind'],
               'admission_verifier_public_key': public_hex(anchor),
               'registry': registry, 'epoch': epoch, 'now': old_fixture['now'],
               'revoked': old_fixture['revoked']}

    write('review_root_grant.json', root)
    write('review_child_grant.json', child)
    write('review_action_proof.json', proof)
    write('review_public_registry.json', fixture)
    return fixture['admission_verifier_public_key']

if __name__ == '__main__':
    anchor = regenerate()
    result = run()
    print(json.dumps({'admission_verifier_public_key': anchor, 'replay_check': result},
                     indent=2))
    raise SystemExit(0 if result['first'] == 'permitted_once_in_local_model'
                     and result['replay'] == 'rejected'
                     and result['local_effect_count'] == 1 else 1)
```

Appendix F. Full policy examples and public signed-record fixtures

The full original policy example is preserved. The new signed records were generated with ephemeral synthetic keys for this review model. Only public keys, certificates, request data, signatures, and synthetic registry bindings are published. No actual human commitment or production key is contained here. The root is signed by a synthetic admission authority, not by an executed Rosario–Wang human ceremony. The timestamps and identifiers are model values. These records are reproducible inputs to run_public_example.py. regenerate_public_example.py (Appendix E.10) rebuilds the root grant, child grant, action proof, and public registry with fresh ephemeral keys from the same request, policies, epoch, and time values, and discards the private keys when it exits. [D2]

The example's trusted root is configured outside the child grant. The child grant binds its preceding record, its own key, policy, measurement, profile, lifetime, epoch, and birth nonce. The action proof binds the exact request and stream position. The receiver independently chooses the expected peer, channel, sequence, and geometry digest from the test fixture. The ledger accepts one modeled effect and rejects a duplicate.

F.1. Authority_Policy_Example.yaml

```
# Illustrative, unadmitted policy. Digest values are example placeholders.
schema_version: http6.policy.v1
policy_id: finance-report-worker-v1
semantics:
  profile: http6-monotone-v1
  default_decision: deny
  conflict_resolution: deny_overrides
  evaluation_error: deny
  ancestor_combiner: conjunction
  canonicalization: RFC8785
  engine_digest: sha256:0000000000000000000000000000000000000000000000000000000000000001
  normalizer_digest: sha256:0000000000000000000000000000000000000000000000000000000000000002
  action_registry_digest: sha256:0000000000000000000000000000000000000000000000000000000000000003
  context_resolver_digest: sha256:0000000000000000000000000000000000000000000000000000000000000004
  operation_adapter_digest: sha256:0000000000000000000000000000000000000000000000000000000000000005
iam:
  tenant: acme
  active_roles:
  - report-reader
  role_catalog_digest: sha256:0000000000000000000000000000000000000000000000000000000000000006
  assignment_snapshot_digest: sha256:0000000000000000000000000000000000000000000000000000000000000007
  role_grants:
  - role: report-reader
    operations:
    - ledger.read
    - report.write
permissions:
  allow:
  - id: read-approved-ledger
    operations:
    - ledger.read
    destinations:
    - ledger-service
    resources:
    - urn:acme:ledger:approved-view
    conditions:
    - field: request.tenant
      operator: eq
      value: acme
  - id: publish-approved-report
    operations:
    - report.write
    destinations:
    - report-service
```

```
    resources:
    - urn:acme:reports:daily
    conditions: []
  deny:
  - id: block-destructive-operations
    operations:
    - ledger.delete
    - iam.grant
    - process.exec
    destinations:
    - '*'
    resources:
    - '*'
    conditions: []
network:
  broker_only: true
  reauthorize_redirects: true
  trust_bundle_digest: sha256:0000000000000000000000000000000000000000000000000000000000000008
  source_route_id: acme-finance-zone-v1
  allow:
  - id: ledger-service
    dns_name: ledger.example.internal
    service_identity: spiffe://acme.example/service/ledger
    ports:
    - 443
    protocols:
    - tls
    ip_cidrs:
    - 10.20.0.0/24
  - id: report-service
    dns_name: reports.example.internal
    service_identity: spiffe://acme.example/service/reports
    ports:
    - 443
    protocols:
    - tls
    ip_cidrs:
    - 10.21.0.0/24
  deny:
    dns_names:
    - unapproved-export.example
    service_identities: []
    ip_cidrs:
    - 169.254.0.0/16
delegation:
  enabled: false
  max_depth: 0
  max_direct_children: 0
  max_total_descendants: 0
  child_template_digests: []
lifecycle:
  root_restart: human_proof
  child_restart: live_parent_fresh_attestation
  route_change: new_admission
  lease_seconds: 60
  max_lifetime_seconds: 3600
runtime:
  launcher_digest: sha256:0000000000000000000000000000000000000000000000000000000000000009
  enforcement_profile_digest: sha256:000000000000000000000000000000000000000000000000000000000000000a
  approved_code_digests:
  - sha256:000000000000000000000000000000000000000000000000000000000000000b
  machine_inventory_digest: sha256:000000000000000000000000000000000000000000000000000000000000000c
  require_fresh_attestation: true
  require_nonexportable_instance_key: true
  inherit_instance_keys: false
  plugin_digests: []
proof:
  profile: http6-computational-v1
  action_body_bound: true
  one_use: true
  channel_bound: true
```

```
  strict_sequence: true
  ledger_namespace: acme-finance-actions
budgets:
  ledger_id: acme-finance-budget
  scope: subtree
  limits:
  - dimension: operations
    maximum: 1000
    window_seconds: 3600
  - dimension: export_bytes
    maximum: 1048576
    window_seconds: 3600
audit:
  event_types:
  - birth
  - decision
  - outcome
  - revoke
  - recovery
  - rejection
  private_reference_logging: false
  retention_seconds: 2592000
```

F.2. review_policy.json

```
{
  "version": "http6.review-policy.v1",
  "tenant": "acme",
  "allow": [
    {
      "operation": "report.publish",
      "destination": "reports.example",
      "resource": "daily",
      "data_class": "aggregate"
    }
  ],
  "deny": [],
  "role_snapshot": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
  "can_mint": true,
  "max_depth": 4,
  "templates": [
    "worker-v1"
  ],
  "budget": 10
}
```

F.3. review_root_grant.json

```
{
  "body": {
    "version": "http6.review-grant.v1",
    "instance": "8dc91d6f941df68238d729bad4cd9cc9dcde035eb255d2ba74d4f5c9338a9ca7",
    "public_key": "04a5ae4ab119fae07be7734e5a15321087f60cfe51f1815ef9875f0aa362ec9c2245618420bf3509fc2d5
    ↪  da27166650e456e2055e3e5eaf7547a71138727b0c6ab",
    "birth_nonce": "d162a17852ae5e944d8978493e134590",
    "parent": null,
    "root": "8dc91d6f941df68238d729bad4cd9cc9dcde035eb255d2ba74d4f5c9338a9ca7",
    "epoch": 42,
    "profile": "ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc",
    "issued": 10,
    "expires": 100,
    "measurement": "root-v1",
    "policy": {
      "version": "http6.review-policy.v1",
      "tenant": "acme",
      "allow": [
        {
          "operation": "report.publish",
          "destination": "reports.example",
```

```
          "resource": "daily",
          "data_class": "aggregate"
        }
      ],
      "deny": [],
      "role_snapshot": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
      "can_mint": true,
      "max_depth": 4,
      "templates": [
        "worker-v1"
      ],
      "budget": 10
    },
    "policy_digest": "9ce324d229e70000fff2bafd7d5899a0e9f5c7d1a06b174633d193a4c9a5b4d0"
  },
  "signature": "3046022100efb225dfdc48bd42949a3260eee3add08bf7bfa9d64c0ae114628025ec984d5a022100d929135e
  ↪  ffe55286979f0e8c2e686eae085722b3083aa0ff950e0f8402420dfb"
}
```

F.4. review_child_grant.json

```
{
  "body": {
    "version": "http6.review-grant.v1",
    "instance": "0a6fcc03c08a71c2632a30257b0080e2d0b0f90820593d5b4ace833cf1ad6477",
    "public_key": "0486e670b9ee353e61e805962137ad1799659380f9ea2fba6cdd58de1fedd4f342ce556439d14ad6ec851
    ↪  a434c53908b53e65a5a94a5ad4045806c7e4b46161d6b",
    "birth_nonce": "e7f4db7822a921e3bf80d2f4d4e01b65",
    "parent": "83fa6f38d854573f8bef03d40179053040555ac1ca542afe1fc5c130eb282bb0",
    "root": "8dc91d6f941df68238d729bad4cd9cc9dcde035eb255d2ba74d4f5c9338a9ca7",
    "epoch": 42,
    "profile": "ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc",
    "issued": 20,
    "expires": 90,
    "measurement": "worker-v1",
    "policy": {
      "version": "http6.review-policy.v1",
      "tenant": "acme",
      "allow": [
        {
          "operation": "report.publish",
          "destination": "reports.example",
          "resource": "daily",
          "data_class": "aggregate"
        }
      ],
      "deny": [],
      "role_snapshot": "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
      "can_mint": false,
      "max_depth": 0,
      "templates": [
        "worker-v1"
      ],
      "budget": 10
    },
    "policy_digest": "f99f87e2fdc5d196f96b37750cda4c9eef7496872e4d73f9b096a322e459a417"
  },
  "signature": "3044022068e78b20d016a7c8ddafda77efede2dc330fe42b1cd18ba4b454496a790bd7bd02206faa20c5a04f
  ↪  a2c33f58f0ce166fac375bbaa55e5fefb26e46fcd0245a9e65d5"
}
```

F.5. review_public_registry.json

```
{
  "fixture_kind": "synthetic_only",
  "admission_verifier_public_key": "0415101e681e289dfc6a85f211e854913ecfa9c804b2405be025ca406e073bc9236c
  ↪  acdac8d584087c9684343ea1f02e596964fc2d2ad27ba4ed1972e29ab59bb0",
  "registry": {
    "8dc91d6f941df68238d729bad4cd9cc9dcde035eb255d2ba74d4f5c9338a9ca7": {
```

```
      "public_key": "04a5ae4ab119fae07be7734e5a15321087f60cfe51f1815ef9875f0aa362ec9c2245618420bf3509fc2
      ↪  d5da27166650e456e2055e3e5eaf7547a71138727b0c6ab",
      "measurement": "root-v1",
      "active": true
    },
    "0a6fcc03c08a71c2632a30257b0080e2d0b0f90820593d5b4ace833cf1ad6477": {
      "public_key": "0486e670b9ee353e61e805962137ad1799659380f9ea2fba6cdd58de1fedd4f342ce556439d14ad6ec8
      ↪  51a434c53908b53e65a5a94a5ad4045806c7e4b46161d6b",
      "measurement": "worker-v1",
      "active": true
    }
  },
  "epoch": 42,
  "now": 25,
  "revoked": []
}
```

F.6. review_request.json

```
{
  "tenant": "acme",
  "operation": "report.publish",
  "destination": "reports.example",
  "resource": "daily",
  "data_class": "aggregate",
  "payload_digest": "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb",
  "cost": 1
}
```

F.7. review_action_proof.json

```
{
  "body": {
    "instance": "0a6fcc03c08a71c2632a30257b0080e2d0b0f90820593d5b4ace833cf1ad6477",
    "epoch": 42,
    "profile": "ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc",
    "policy_digest": "f99f87e2fdc5d196f96b37750cda4c9eef7496872e4d73f9b096a322e459a417",
    "lineage_digest": "47dbc829532d56dd4da38122e74f91d1e9865d0f4119603373e80dda80390fb2",
    "channel": "channel-1",
    "sequence": 0,
    "geometry_digest": "ddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd",
    "nonce": "nonce-1",
    "expires": 70,
    "direction": "request",
    "request": {
      "tenant": "acme",
      "operation": "report.publish",
      "destination": "reports.example",
      "resource": "daily",
      "data_class": "aggregate",
      "payload_digest": "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb",
      "cost": 1
    }
  },
  "signature": "3046022100e5e978238c2d2bde27d518d300cd616f8f80f1b6cce2c07d9d3923fb14accefa022100f23c5a99
  ↪  987b9ab4005f24815cf5963997dbc84c47819ef678717b9d79afe76d"
}
```

F.8. review_evidence_record.json

```
{
  "requirement_id": "NIST-SP800-53r5:IA-2(8)",
  "scope_id": "illustrative-reporting-enclave",
  "source_locator": "IA-2(8)",
  "implementation_owner": "identity-service-owner",
  "implementation_ref": "admission-gate-proposal-v1",
  "methods": [
    "examine",
```

```
    "test"
  ],
  "expected": "One accepted consumption for each admitted position",
  "observed": null,
  "evidence_digests": [],
  "status": "not_assessed",
  "reviewer": null
}
```

F.9. canonical_test_vector.json

```
{
  "profile": "TEST-CANON-1",
  "input": {
    "b": 2,
    "a": 1
  },
  "canonical_utf8": "{\"a\":1,\"b\":2}",
  "sha256": "43258cff783fe7036d8a43033f830adfc60ec037382473548ac742b888292777"
}
```

Appendix G. Complete raw examples for the twenty control families

All twenty control families appear below. The named control identifiers are **selected anchors within each family**, not a claim that these examples enumerate every control or enhancement. Each family has a concrete problem scenario, source-based mechanism or explicit integration, raw case, acceptance behavior and completion evidence. The authoritative catalog and selected baseline supply the full clause inventory under Chapters 9–11 and Appendix J. NIST control identifiers and family names come from the selected catalog; the scenarios and HTTP6 mappings are original engineering examples. [N1, N2, N16]

G.1. AC — Access Control

Selected NIST anchors: AC-2, AC-3, AC-4, AC-5, AC-6, AC-7, AC-12, AC-16, AC-20, AC-25. **Source relationship:** [S1, II.5; II.6.5–II.6.6; III.5–III.17]. **Implementation mode:** source_mechanism_plus_integration. [N1]

Problem illustrated. A reporting worker has legitimate ledger access but is instructed to export the complete customer table to an unrelated host. A permissive intermediary could otherwise turn its own broad service account into an escape from the worker's restrictions.

Architecture response. The resource gate evaluates the complete operation, object, payload class, authenticated destination, current identity, and every ancestor policy. The child's effective permission is the intersection of the root ceiling, scheduler grant, and worker policy. An explicit deny wins over an overlapping allow. Two-person approval is a separate integration with two independently authenticated human principals; creating a child does not produce a second approver.

```
example_id: EX-AC-01
fixture_status: illustrative_not_executed
selected_controls:
- AC-2
- AC-3
- AC-4
- AC-5
- AC-6
- AC-7
- AC-12
- AC-16
- AC-20
- AC-25
implementation_mode: source_mechanism_plus_integration
owner: access-control-owner
input:
  root: finance-root/e42
  child: report-worker/17
  allow:
    operation: report.publish
    destination: reports.example.internal
    data_class: approved-aggregate
  deny:
    operation: ledger.export
    destination: external.example
  request:
    operation: ledger.export
    destination: external.example
    data_class: raw-customer
expected_positive: An approved aggregate reaches the report service through the same gate and consumes
  one action position.
expected_adverse: The raw export is denied at both the egress broker and destination-facing resource
↪  adapter;
  no broader broker identity replaces the caller's scope.
observed_result: null
evidence_required:
- approved role assignments
```

```
- ancestor policies and canonical digests
- positive and negative request traces
- independent resource-side effect inspection
```

Evidence and accountable completion. AC policies also require account administration, review cadence, privilege approval, access removal, and organization-defined parameters. The cryptographic gate implements decisions; the accountable owner establishes which grants are legitimate and reviews them.

G.2. AT — Awareness and Training

Selected NIST anchors: AT-1, AT-2, AT-3, AT-4. **Source relationship:** [S1, III.7–III.10; III.19]. **Implementation mode:** proposed_governance_integration. [N1]

Problem illustrated. An administrator can complete a cryptographic ceremony while lacking the role-specific knowledge needed to approve a fleet safely. Authentication alone does not teach incident handling or safe delegation.

Architecture response. A trusted learning-system assertion identifies the human, curriculum version, completion date, and role-specific training requirement. The root-admission policy requires a current assertion before issuing high-impact authority. Training content explains trusted confirmation, prohibited recovery shortcuts, phishing reporting, and the distinction between a human ceremony and ongoing machine activity. Records retain the actual course and assessment artifacts, not merely a policy Boolean.

```
example_id: EX-AT-01
fixture_status: illustrative_not_executed
selected_controls:
- AT-1
- AT-2
- AT-3
- AT-4
implementation_mode: proposed_governance_integration
owner: training-program-owner
input:
  subject: human/admin-7
  requested_role: fleet-root-approver
  training_assertion:
    issuer: approved-learning-service
    course: root-operations/v3
    valid: false
  root_admission: requested
expected_positive: A trained administrator with a current verified record can initiate an otherwise
↪  approved
  root ceremony.
expected_adverse: A missing, forged, or expired training record holds administrative admission and directs
  the person to the approved training process.
observed_result: null
evidence_required:
- approved curriculum
- delivery and completion records
- role-specific exercise results
- training-record issuer validation tests
```

Evidence and accountable completion. Training personnel remain responsible for instruction, accessibility, effectiveness measurement, refreshers, and records. The Policy Decision Point adds an enforceable eligibility connection and accountable evidence trail; the people program supplies the learning outcome.

G.3. AU — Audit and Accountability

Selected NIST anchors: AU-2, AU-3, AU-5, AU-6, AU-8, AU-9, AU-11, AU-12. **Source relationship:** [S1, II.5.2; III.19]. **Implementation mode:** source_mechanism_plus_integration. [N1]

Problem illustrated. A valid action leaves a log that identifies only a shared service account, or the worker can erase the evidence of its own prohibited request.

Architecture response. The protected collector records the human admission receipt, root epoch, parent and child identities, policy commitment, normalized operation digest, decision, and durable transaction outcome. A worker is allowed to submit events but not edit retained records. Independent storage permissions, authenticated checkpoints, time synchronization, and collection health checks make the trail useful for review. Secret references and active keys are excluded from ordinary logs.

```
example_id: EX-AU-01
fixture_status: illustrative_not_executed
selected_controls:
- AU-2
- AU-3
- AU-5
- AU-6
- AU-8
- AU-9
- AU-11
- AU-12
implementation_mode: source_mechanism_plus_integration
owner: audit-service-owner
input:
  event_id: audit-0017
  root_epoch: 42
  instance: report-worker/17
  operation: ledger.export
  decision: deny
  policy_ref: policy/report-worker/v4
  collector_role: append-only
  worker_request: delete_audit_record
expected_positive: Authorized reviewers reconstruct an approved report from birth through effect and
↪  verify
  record provenance.
expected_adverse: The worker's deletion request fails, and deliberate collection interruption raises the
  configured audit-failure response.
observed_result: null
evidence_required:
- event-selection policy
- protected sample records
- collector outage exercise
- retention and reviewer-access configuration
```

Evidence and accountable completion. A cryptographic chain is one record-integrity mechanism. Storage availability, retention, review staffing, alert disposition, timestamp quality, and access administration complete the audit service. A signature on a fabricated event cannot establish that the represented real-world action occurred.

G.4. CA — Assessment, Authorization, and Monitoring

Selected NIST anchors: CA-2, CA-5, CA-6, CA-7, CA-9. **Source relationship:** [S1, II.8; III.20]. **Implementation mode:** proposed_assessment_integration. [N1]

Problem illustrated. A diagram is treated as proof that every implemented resource enforces the diagram, even though a direct database credential bypasses the protected broker.

Architecture response. The assessment plan names the boundary, applicable clauses, organization-defined parameters, expected behaviors, and methods. An independent test ac-

count attempts each effect through the documented and alternative interfaces. Findings bind to the exact code, policy, adapter, and hardware versions. The release gate uses assessment dispositions and remediation approvals; the designated authorizing official makes the risk decision.

```
example_id: EX-CA-01
fixture_status: illustrative_not_executed
selected_controls:
- CA-2
- CA-5
- CA-6
- CA-7
- CA-9
implementation_mode: proposed_assessment_integration
owner: independent-assessment-lead
input:
  assessment: campaign-CA-01
  boundary: reporting-enclave
  test: direct-db-bypass
  expected: deny
  observed: null
  finding_state: not_run
  authorization_decision: null
expected_positive: A complete reviewed evidence package supports an explicit authorization decision for
  the measured boundary.
expected_adverse: A missing test result or unexplained bypass prevents the implementation from being
↪  reported
  as verified.
observed_result: null
evidence_required:
- assessment plan
- examine-interview-test records
- plan of action and milestones
- signed authorization decision and monitoring plan
```

Evidence and accountable completion. The HTTP6 reference architecture provides traceable objects for assessment. Assessors and officials supply judgments, independence, risk acceptance, and ongoing oversight. Classification presence and passing a demonstration are separate from the full assessment of selected control objectives.

G.5. CM — Configuration Management

Selected NIST anchors: CM-2, CM-3, CM-5, CM-6, CM-7, CM-8, CM-10, CM-11, CM-14. **Source relationship:** [S1, II.6.3; III.7; III.9; III.13; III.18]. **Implementation mode:** source_mechanism_plus_integration. [N1]

Problem illustrated. An unchanged child policy name silently picks up a broader role definition, or a patched normalizer interprets a formerly denied destination as allowed.

Architecture response. The admission manifest pins executable, dependencies, configuration, canonical policy, evaluator, role catalog, action registry, and normalizer digests. A live instance retains that meaning. An approved patch produces a new measured generation and controlled transition. Inventory links source/build provenance to deployed instances and distinguishes quarantined, active, retired, and replaced generations.

```
example_id: EX-CM-01
fixture_status: illustrative_not_executed
selected_controls:
- CM-2
- CM-3
- CM-5
- CM-6
- CM-7
- CM-8
- CM-10
```

```
- CM-11
- CM-14
implementation_mode: source_mechanism_plus_integration
owner: configuration-manager
input:
  instance: worker/17
  admitted_role_catalog: roles/v4
  admitted_normalizer: normalizer/v2
  local_change:
    role_catalog: roles/v5-broader
  requested_action: reuse_old_admission
expected_positive: An approved replacement with matching measurements receives a new admission under the
  existing root ceiling.
expected_adverse: The altered role catalog fails the admitted commitment and cannot expand the existing
  child's authority.
observed_result: null
evidence_required:
- approved baseline manifest
- change request and approvers
- artifact digest comparison
- rollout and rollback test results
```

Evidence and accountable completion. Owners manage inventories, approved configuration values, licenses, installation permissions, emergency changes, and update windows. Immutability preserves identity during a generation; controlled replacement keeps the system maintainable and patched.

G.6. CP — Contingency Planning

Selected NIST anchors: CP-2, CP-3, CP-4, CP-6, CP-7, CP-8, CP-9, CP-10. **Source relationship:** [S1, II.6.2; II.7.3; III.18]. **Implementation mode:** source_mechanism_plus_operational_extension. [N1]

Problem illustrated. A failed root is restored from an old image and outstanding action positions become reusable, or fail-closed behavior protects data while the organization has no practicable recovery path.

Architecture response. Backups separately preserve business outcomes, consumed positions, policies, enrollment administration, and recovery evidence. A living parent may admit an eligible replacement child. A stopped root follows the approved human re-ignition process and establishes a new epoch. Recovery custodians, alternate sites, communications, power, and tested restoration procedures complete the continuity design.

```
example_id: EX-CP-01
fixture_status: illustrative_not_executed
selected_controls:
- CP-2
- CP-3
- CP-4
- CP-6
- CP-7
- CP-8
- CP-9
- CP-10
implementation_mode: source_mechanism_plus_operational_extension
owner: continuity-manager
input:
  incident: root-power-loss
  retired_epoch: 42
  backup_epoch: 41
  recovery_request: resume_epoch_41
  approved_path: new_human_ceremony_then_epoch_43
  business_operation: report-2026-1109
expected_positive: The new epoch resumes the authorized task while preserving the previously recorded
  transaction outcome.
expected_adverse: The obsolete snapshot cannot reset nonce consumption or impersonate a currently admitted
  root.
```

```
observed_result: null
evidence_required:
- business impact analysis
- backup integrity and restore records
- recovery role availability exercise
- measured recovery-time and data-loss observations
```

Evidence and accountable completion. Recovery objectives are organization-selected and measured. Hardware isolation and spare batteries support survivability, but backup completeness, staffing, alternate infrastructure, and exercises supply the operational capability.

G.7. IA — Identification and Authentication

Selected NIST anchors: IA-2, IA-2(8), IA-3, IA-4, IA-5, IA-8, IA-9, IA-11, IA-12. **Source relationship:** [S1, II.3–II.7; III.2–III.4]. **Implementation mode:** source_mechanism_plus_identity_service_integration. [N1]

Problem illustrated. A recorded human response is replayed at another verifier, or a workload identity is confused with a fresh interaction by the human who originally approved it.

Architecture response. The Human Admission Client collects the human ceremony through the protected interaction path. Rosario–Wang verification evaluates the enrolled relation; the Proof Authority binds acceptance to verifier, channel, policy, instance key, epoch, and one-use challenge. Children authenticate as distinct workloads. Proofing, account binding, authenticator management, and independent human reauthentication are supplied by the selected identity profile.

```
example_id: EX-IA-01
fixture_status: illustrative_not_executed
selected_controls:
- IA-2
- IA-2(8)
- IA-3
- IA-4
- IA-5
- IA-8
- IA-9
- IA-11
- IA-12
implementation_mode: source_mechanism_plus_identity_service_integration
owner: identity-service-owner
input:
  human_auth_time: '2026-11-09T17:00:00Z'
  root_epoch: 42
  nonce: challenge-19
  audience: ledger.example.internal
  ledger_state: consumed
  attack:
    reuse_nonce: challenge-19
    audience: lookalike.example
expected_positive: A fresh ceremony bound to the genuine verifier activates only the named instance and
  policy.
expected_adverse: The old nonce or different audience fails; machine heartbeats never change
↪  human_auth_time.
observed_result: null
evidence_required:
- enrollment and binding records
- authenticator profile and key-custody evidence
- cross-origin and replay tests
- human timeout and recovery tests
```

Evidence and accountable completion. Assurance is assigned to the complete selected authentication transaction. Human proofing records, supported authenticator types, factor independence, session rules, and notification procedures are evaluated separately from geometric complexity.

G.8. IR — Incident Response

Selected NIST anchors: IR-2, IR-3, IR-4, IR-5, IR-6, IR-7, IR-8. **Source relationship:** [S1, III.14–III.20]. **Implementation mode:** source_mechanism_plus_response_program. [N1]

Problem illustrated. One compromised worker remains usable because response removes its local process but not its remote leases, siblings, or issued descendants.

Architecture response. The registry traces the affected lineage. An authorized responder fences the chosen child, subtree, or root epoch, freezes relevant evidence, and stops new issuance and renewal. Resource gates enforce the fence. Human review, triage, severity, reporting contacts, eradication, and recovery criteria reside in the response plan. The workflow follows the CSF-aligned incident lifecycle described by the selected incident-response guidance.

```
example_id: EX-IR-01
fixture_status: illustrative_not_executed
selected_controls:
- IR-2
- IR-3
- IR-4
- IR-5
- IR-6
- IR-7
- IR-8
implementation_mode: source_mechanism_plus_response_program
owner: incident-commander
input:
  incident_id: IR-17
  affected_parent: scheduler/2
  response: fence_subtree
  preserve:
  - birth-records
  - action-ledger
  - approved-audit-snapshot
  new_child_request: pending
expected_positive: Unaffected separately authorized work continues when the containment decision permits
  it.
expected_adverse: The fenced parent cannot issue a fresh child; all affected gates enforce revocation
  within the documented liveness bound.
observed_result: null
evidence_required:
- approved incident plan
- triage and escalation records
- resource-side fence tests
- evidence custody and recovery authorization records
```

Evidence and accountable completion. Notification deadlines and recipients come from the organization's obligations, not arbitrary example values. SP 800-61 Revision 3 supplies updated incident-response context; operational drills establish whether staff and service integrations perform the plan.

G.9. MA — Maintenance

Selected NIST anchors: MA-2, MA-3, MA-4, MA-5, MA-6. **Source relationship:** [S1, III.9–III.13; III.18]. **Implementation mode:** proposed_maintenance_integration. [N1]

Problem illustrated. A vendor receives unrestricted shell access to repair a server and retains a reusable administrative credential after the service window.

Architecture response. A maintenance circuit is admitted under a ticket, approved technician identity, assessed maintenance tool, exact target, permitted commands, and bounded time. Its policy prohibits unrelated data export and child creation. Remote access uses the same fresh-proof

gate; local servicing additionally follows physical escort and equipment-handling procedures. Completion retires the maintenance generation and triggers integrity re-appraisal.

```
example_id: EX-MA-01
fixture_status: illustrative_not_executed
selected_controls:
- MA-2
- MA-3
- MA-4
- MA-5
- MA-6
implementation_mode: proposed_maintenance_integration
owner: maintenance-authority
input:
  ticket: MA-58
  technician: vendor/approved-4
  target: host/23
  allow:
  - health.inspect
  - approved_patch.apply
  deny:
  - ledger.export
  - iam.grant
  - child.spawn
  expires_at: '2026-11-09T19:00:00Z'
expected_positive: The approved tool performs the ticketed operation during the authorized window and
  records its outcome.
expected_adverse: An unrelated command or request after expiry is denied even when the technician retains
  a copied ticket.
observed_result: null
evidence_required:
- maintenance ticket and approvers
- tool validation record
- technician authorization
- session termination and post-service measurement
```

Evidence and accountable completion. Maintenance scheduling, spare parts, technician supervision, and local physical procedures have operational owners. The circuit constrains the digital privileges used in performing that work.

G.10. MP — Media Protection

Selected NIST anchors: MP-2, MP-3, MP-4, MP-5, MP-6, MP-7. **Source relationship:** [S1, II.6.6; III.12; III.15]. **Implementation mode:** proposed_media_lifecycle_integration. [N1]

Problem illustrated. A retired storage device or exported backup retains sensitive data, even though its online account has been revoked.

Architecture response. Media identity, classification, location, transport custodian, and disposal state are registered. Export requires an authorized data-flow operation. Sanitization is a separately approved job with device-specific method, verification, and disposition records. Controlled key release protects encrypted media while active; cryptographic erase is selected only when the full key-copy, backup, and storage design supports that method.

```
example_id: EX-MP-01
fixture_status: illustrative_not_executed
selected_controls:
- MP-2
- MP-3
- MP-4
- MP-5
- MP-6
- MP-7
implementation_mode: proposed_media_lifecycle_integration
owner: media-custodian
input:
  media_id: drive-204
```

```
  classification: restricted
  disposition: retire
  sanitization_method: organization-approved-device-specific-method
  verification_record: null
  release_for_reuse: false
expected_positive: Authorized disposition follows the selected sanitization process and verified evidence
  before release.
expected_adverse: A revoked network key alone does not mark the physical media sanitized; missing
↪  verification
  holds release.
observed_result: null
evidence_required:
- media inventory and custody
- approved sanitization procedure
- device-specific verification record
- disposition approval
```

Evidence and accountable completion. SP 800-88 Revision 2 is the selected sanitization guidance. Physical destruction, transport safeguards, labeling, handling, and verification are performed by responsible personnel or equipment; the Policy Decision Point authorizes them and the audit trail records them rather than claiming to perform a physical action through a hash.

G.11. PE — Physical and Environmental Protection

Selected NIST anchors: PE-2, PE-3, PE-6, PE-8, PE-11, PE-13, PE-14, PE-15, PE-16. **Source relationship:** [S1, II.6.3; III.10; III.18]. **Implementation mode:** proposed_facility_and_hardware_integrati on. [N1]

Problem illustrated. A well-authenticated root runs in a room with uncontrolled entry, unreliable power, or no tested response to fire and cooling failure.

Architecture response. The facility supplies independently enforced locks, visitor procedures, monitoring, power, fire protection, and environmental controls. The Policy Decision Point can require current approved hosting evidence before releasing sensitive authority, and the Authoritative State Plane can fence affected authority on authenticated tamper or environment alarms. Facility events are bound to the affected machine inventory and incident records. Life-safety controls and emergency egress remain governed by their appropriate systems.

```
example_id: EX-PE-01
fixture_status: illustrative_not_executed
selected_controls:
- PE-2
- PE-3
- PE-6
- PE-8
- PE-11
- PE-13
- PE-14
- PE-15
- PE-16
implementation_mode: proposed_facility_and_hardware_integration
owner: facilities-security-owner
input:
  asset: root-host/1
  site: approved-room/A
  event:
    kind: tamper-alarm
    issuer: registered-facility-controller
  policy_action: suspend_new_key_release
  life_safety_override: independent_facility_procedure
expected_positive: A permitted operator enters under the facility's physical process and the hosting
↪  evidence
  remains current.
expected_adverse: A cloned digital badge record does not authorize door entry; a validated host alarm
  prevents new sensitive admissions pending review.
```

```
observed_result: null
evidence_required:
- access roster and visitor records
- alarm and response exercise
- power/failover maintenance records
- facility control-provider assessment
```

Evidence and accountable completion. A TEE supports a specific runtime protection boundary; it does not itself provide fire suppression, secure delivery, visitor escort, or emergency power. Those services receive named control-provider responsibilities and evidence dependencies.

G.12. PL — Planning

Selected NIST anchors: PL-2, PL-4, PL-8, PL-9, PL-10, PL-11. **Source relationship:** [S1, III.8–III.10; III.20]. **Implementation mode:** proposed_security_planning_integration. [N1]

Problem illustrated. The deployment protects one broker but the documented system boundary omits a backup export account and an administrative database route.

Architecture response. The security and privacy plans enumerate processes, resources, trust boundaries, data flows, inherited services, users, baseline selection, and implementation responsibilities. A signed deployment profile identifies the mandatory gates and approved adapters. Architecture review reconciles the inventory with observed routes. Rules of behavior are communicated and acknowledged through a separate human process.

```
example_id: EX-PL-01
fixture_status: illustrative_not_executed
selected_controls:
- PL-2
- PL-4
- PL-8
- PL-9
- PL-10
- PL-11
implementation_mode: proposed_security_planning_integration
owner: system-security-plan-owner
input:
  system_plan: SSP-reporting/v2
  protected_resources:
  - ledger
  - report-store
  - backup-store
  required_adapters:
  - db-gate
  - egress-gate
  - backup-gate
  discovered_route: unmediated-backup-export
expected_positive: Every usable route in the accepted boundary is assigned to an approved enforcement
  point and owner.
expected_adverse: The unmediated route opens a planning and implementation finding rather than
↪  disappearing
  from the coverage denominator.
observed_result: null
evidence_required:
- current system security and privacy plans
- boundary and data-flow review
- user rules acknowledgment
- control selection and tailoring record
```

Evidence and accountable completion. The planning owner keeps records aligned with real changes. Cryptographic policy generation depends on a correct boundary inventory; a perfectly signed incomplete plan remains incomplete.

G.13. PM — Program Management

Selected NIST anchors: PM-1, PM-2, PM-3, PM-5, PM-6, PM-9, PM-13, PM-14, PM-18. **Source relationship:** [S1, III.1; III.19–III.20]. **Implementation mode:** proposed_enterprise_governance_integration. [N1]

Problem illustrated. The product has a traceable human administrator but no approved security program, funding, workforce plan, or management review of systemic risk.

Architecture response. Executive-approved program records name decision authorities, security and privacy leadership, resources, inventory responsibility, metrics, risk treatment, and review cadence. The Proof Authority links technical root ownership to that organizational authority rather than assuming that whoever can ignite a root is authorized to set enterprise policy. Management decisions are retained as evidence and drive controlled profile changes.

```
example_id: EX-PM-01
fixture_status: illustrative_not_executed
selected_controls:
- PM-1
- PM-2
- PM-3
- PM-5
- PM-6
- PM-9
- PM-13
- PM-14
- PM-18
implementation_mode: proposed_enterprise_governance_integration
owner: executive-risk-owner
input:
  program: enterprise-security/v3
  accountable_role: executive-risk-owner
  root_policy_owner: finance-system-owner
  budget_approval: approval-record-PM-8
  review_due: '2027-02-01'
  unsigned_policy_expansion: requested
expected_positive: An authorized governance decision can initiate a properly approved new root policy
  and assessment scope.
expected_adverse: A technically capable administrator cannot impersonate the management approval needed
  for enterprise scope expansion.
observed_result: null
evidence_required:
- approved program plan
- role appointments and resources
- risk register and review minutes
- program metrics and corrective-action records
```

Evidence and accountable completion. Program controls are organization-wide activities. The Policy Decision Point supplies accountability and policy enforcement within them; executive leadership, resource allocation, and sustained management practice establish the program itself.

G.14. PS — Personnel Security

Selected NIST anchors: PS-2, PS-3, PS-4, PS-5, PS-6, PS-7, PS-8. **Source relationship:** [S1, III.7; III.18–III.19]. **Implementation mode:** proposed_personnel_integration. [N1]

Problem illustrated. An employee leaves the organization while previously spawned agents continue receiving renewals from a forgotten root.

Architecture response. Trusted personnel events link position requirements, screening, access agreements, transfers, and termination to the identity service. A termination or role-removal event fences the affected admissions and prevents renewal, while retained lineage records support re-

view. Reassignment uses newly approved role snapshots. The human-resource service authenticates the event and the access owner confirms the affected account relationships.

```
example_id: EX-PS-01
fixture_status: illustrative_not_executed
selected_controls:
- PS-2
- PS-3
- PS-4
- PS-5
- PS-6
- PS-7
- PS-8
implementation_mode: proposed_personnel_integration
owner: personnel-security-owner
input:
  person: human/admin-7
  personnel_event: termination
  event_issuer: approved-personnel-service
  affected_root: finance-root/e42
  expected_action: revoke_root_and_descendant_renewal
expected_positive: An authorized role transfer receives a reviewed narrower admission appropriate to the
  new duties.
expected_adverse: An old worker cannot keep authority alive by replaying a pre-termination eligibility
  assertion.
observed_result: null
evidence_required:
- position and screening records
- access agreements
- verified termination event
- account and descendant revocation review
```

Evidence and accountable completion. Employment screening, sanctions, legal agreements, and coordination with third-party personnel are performed by the organization. The Authoritative State Plane connects those verified decisions to machine authority and demonstrates that access removal propagated.

G.15. PT — Personally Identifiable Information Processing and Transparency

Selected NIST anchors: PT-2, PT-3, PT-4, PT-5, PT-6, PT-7. **Source relationship:** [S1, II.4; II.7.1; III.12; III.19]. **Implementation mode:** source_privacy_mechanism_plus_privacy_program. [N1]

Problem illustrated. A technically authorized analytics circuit uses authentication or customer data for a purpose that the organization has not authorized or disclosed.

Architecture response. The policy binds permitted data categories, processing purpose, recipients, retention rules, and applicable consent or other authorization records. Privacy review establishes the legal and organizational basis outside the proof engine. The engine enforces approved use and logs minimized evidence. The transcript-privacy proposition protects a specified secret under its premises; purpose limitation and transparency cover other personal information, including linkable genealogy metadata.

```
example_id: EX-PT-01
fixture_status: illustrative_not_executed
selected_controls:
- PT-2
- PT-3
- PT-4
- PT-5
- PT-6
- PT-7
implementation_mode: source_privacy_mechanism_plus_privacy_program
owner: privacy-officer
input:
  dataset: customer-contact
  approved_purpose: account-recovery
```

```
  requested_purpose: marketing-enrichment
  consent_receipt: null
  destination: analytics-service
expected_positive: A properly authorized recovery operation receives only the attributes required for
  its purpose.
expected_adverse: The secondary-purpose request is denied even though the analytics service has a valid
  instance identity.
observed_result: null
evidence_required:
- processing-purpose inventory
- privacy notices and authorization records
- minimization and retention tests
- privacy-risk review and applicable reporting records
```

Evidence and accountable completion. Consent is used where appropriate and is not invented as the universal legal basis. Applicable notice, records, reporting, sensitive-category, and individual-rights duties retain privacy-office owners. Geometry does not erase those obligations.

G.16. RA — Risk Assessment

Selected NIST anchors: RA-2, RA-3, RA-5, RA-7, RA-8, RA-9. **Source relationship:** [S1, II.8; III.20]. **Implementation mode:** proposed_risk_management_integration. [N1]

Problem illustrated. A team treats a large geometric key representation as a complete answer to business impact, exploitable software defects, privacy harms, and operational exposure.

Architecture response. The risk register connects assets and authorization paths to threat scenarios, likelihood assumptions, impacts, vulnerabilities, dependencies, and response decisions. Threat modeling examines private-reference exposure, verifier compromise, bypass routes, concurrency, denial of service, and policy-adapter errors separately. Risk findings change selected requirements and tests; approved treatment may narrow a policy, replace a component, add monitoring, or fund resilience.

```
example_id: EX-RA-01
fixture_status: illustrative_not_executed
selected_controls:
- RA-2
- RA-3
- RA-5
- RA-7
- RA-8
- RA-9
implementation_mode: proposed_risk_management_integration
owner: risk-assessment-owner
input:
  risk_id: RA-broker-01
  scenario: direct-db-route-bypasses-lineage-gate
  asset: ledger
  business_impact: organization-assessed
  treatment: remove_route_and_retest
  owner: data-platform-owner
  closure_evidence: null
expected_positive: Treatment is verified against the affected route and the assessment records the residual
  risk decision.
expected_adverse: A risk is not closed solely because another layer has a strong mathematical proposition.
observed_result: null
evidence_required:
- categorization rationale
- threat and privacy-risk assessments
- vulnerability and supplier findings
- approved treatment and retest evidence
```

Evidence and accountable completion. The supplied theory identifies useful security invariants. Risk assessment determines whether those invariants and supporting controls address the actual system's consequential failure scenarios.

G.17. SA — System and Services Acquisition

Selected NIST anchors: SA-3, SA-4, SA-5, SA-8, SA-9, SA-10, SA-11, SA-15, SA-15(13), SA-24. **Source relationship:** [S1, III.9–III.13; III.20]. **Implementation mode:** source_admission_controls_plus_acquisition_extension. [N1]

Problem illustrated. A correctly signed component is purchased or deployed without security requirements, reliable build evidence, testing obligations, or a resilient upgrade plan.

Architecture response. Acquisition specifications name trust boundaries, approved cryptographic services, instance isolation, audit syntax, update behavior, and assessment deliverables. Development and release pipelines bind approved source, dependencies, tests, and manifests. The protected launcher admits only the approved artifact. Software assurance evaluates the artifact's behavior rather than treating its signature as proof that it is free from defects.

```
example_id: EX-SA-01
fixture_status: illustrative_not_executed
selected_controls:
- SA-3
- SA-4
- SA-5
- SA-8
- SA-9
- SA-10
- SA-11
- SA-15
- SA-15(13)
- SA-24
implementation_mode: source_admission_controls_plus_acquisition_extension
owner: acquisition-and-development-owner
input:
  release: policy-engine/v6
  required_deliverables:
  - security-requirements
  - test-results
  - dependency-manifest
  - update-plan
  - logging-contract
  received:
  - dependency-manifest
  - signature
  admission_state: hold
expected_positive: A reviewed release package can progress through staged replacement with regression
  evidence.
expected_adverse: The signed but incomplete acquisition package is held until its required assurance
↪  deliverables
  are provided and accepted.
observed_result: null
evidence_required:
- contract security requirements
- development and design review records
- independent and developer testing
- approved release and resilience/update evidence
```

Evidence and accountable completion. The 5.2.0 catalog additions include SA-15(13) and SA-24. Their mapping is developed here as logging-contract and resilience engineering, not as a claim that the source manuscripts already contain a completed acquisition program.

G.18. SC — System and Communications Protection

Selected NIST anchors: SC-3, SC-5, SC-6, SC-7, SC-7(5), SC-8, SC-12, SC-13, SC-23, SC-24, SC-28, SC-39. **Source relationship:** [S1, II.5–II.6; III.12–III.16]. **Implementation mode:** source_mechanism_plus_platform_and_crypto_integration. [N1]

Problem illustrated. An authenticated session is redirected to an unauthorized endpoint, or applications bypass the broker through raw sockets while sharing reusable secrets.

Architecture response. Twins bind peers, channels, geometry state, sequence, operation, and policy. The egress broker authenticates the actual endpoint and reauthorizes redirects. Protected key services release or use keys only for permitted instances. Runtime isolation and resource-side gates prevent ambient credentials from becoming independent authority. Capacity controls, rate limits, and shared-budget conservation complement confidentiality and integrity.

```
example_id: EX-SC-01
fixture_status: illustrative_not_executed
selected_controls:
- SC-3
- SC-5
- SC-6
- SC-7
- SC-7(5)
- SC-8
- SC-12
- SC-13
- SC-23
- SC-24
- SC-28
- SC-39
implementation_mode: source_mechanism_plus_platform_and_crypto_integration
owner: platform-security-owner
input:
  request: report.publish
  allowed_peer: reports.example.internal
  redirect_peer: external.example
  channel_generation: 3
  policy_digest: fixture-policy-4
  redirect_authorization: required
expected_positive: A fresh correctly bound request reaches the approved peer and verifies the response
  direction and sequence.
expected_adverse: The redirect or reused channel record fails before the unauthorized endpoint receives
  sensitive payload.
observed_result: null
evidence_required:
- network and crypto profile
- peer/channel binding tests
- broker-bypass and process-isolation tests
- resource and denial-of-service capacity exercises
```

Evidence and accountable completion. Protected channels, storage encryption, approved key services, isolation, and availability mechanisms each need their own implementation evidence. High-dimensional representation is not a substitute for a specified cryptographic mode or infrastructure capacity.

G.19. SI — System and Information Integrity

Selected NIST anchors: SI-2, SI-2(7), SI-3, SI-4, SI-5, SI-7, SI-7(12), SI-10, SI-11. **Source relationship:** [S1, II.6.3; III.9; III.13–III.20]. **Implementation mode:** source_integrity_mechanisms_plus_defect_management. [N1]

Problem illustrated. A measured application remains vulnerable after a defect is discovered, or malformed arguments exploit a mismatch between policy validation and the actual service operation.

Architecture response. Typed inputs, canonical operation encoding, and pinned adapters ensure the approved action matches the executed effect. Integrity checks detect altered artifacts and monitoring identifies suspicious behavior. The vulnerability program validates reports, prioritizes re-

mediation, analyzes root causes, tests fixes, and admits replacement generations. Error messages expose controlled information while internal records retain diagnostic detail.

```
example_id: EX-SI-01
fixture_status: illustrative_not_executed
selected_controls:
- SI-2
- SI-2(7)
- SI-3
- SI-4
- SI-5
- SI-7
- SI-7(12)
- SI-10
- SI-11
implementation_mode: source_integrity_mechanisms_plus_defect_management
owner: integrity-and-vulnerability-owner
input:
  finding: normalizer-double-decode
  affected_version: adapter/v2
  request_path: ambiguous-encoded-path
  expected: reject-before-execution
  remediation: adapter/v3-with-regression-fixture
expected_positive: The fixed adapter passes both ordinary operation and regression cases before controlled
  admission.
expected_adverse: The malformed path is rejected consistently; a signed vulnerable predecessor cannot
  bypass its revocation state.
observed_result: null
evidence_required:
- input and error-handling tests
- monitoring and integrity records
- vulnerability triage and root-cause analysis
- patch validation and rollback tests
```

Evidence and accountable completion. SI-2(7) and the revised SI-7(12) receive explicit release-profile attention. Admission provides integrity and provenance; defect analysis, malicious-code controls, alerts, and remediation supply additional integrity outcomes.

G.20. SR — Supply Chain Risk Management

Selected NIST anchors: SR-2, SR-3, SR-5, SR-6, SR-8, SR-9, SR-10, SR-11, SR-12. **Source relationship:** [S1, III.9–III.13; III.20]. **Implementation mode:** source_provenance_mechanisms_plus_supplier_program. [N1]

Problem illustrated. An untrusted supplier component enters the signing pipeline, or a formerly approved supplier keeps a standing maintenance route after its relationship ends.

Architecture response. The supplier program records provenance, criticality, acquisition review, contractual safeguards, notification arrangements, inspection, authenticity, and disposal requirements. The Minting and Delegation Service admits artifacts and service routes only from the selected authorized suppliers and binds maintenance authority to reviewed tickets. Supplier termination fences related identities, routes, and templates; the organization verifies completion of off-boarding and data-return duties.

```
example_id: EX-SR-01
fixture_status: illustrative_not_executed
selected_controls:
- SR-2
- SR-3
- SR-5
- SR-6
- SR-8
- SR-9
- SR-10
- SR-11
- SR-12
```

```
implementation_mode: source_provenance_mechanisms_plus_supplier_program
owner: supplier-risk-owner
input:
  supplier: vendor/9
  component: runtime-addon
  provenance_status: unverified
  approved_supplier: false
  signature_present: true
  requested_action: admit_to_root_host
expected_positive: A reviewed supplier artifact with accepted provenance and required tests can receive
  admission.
expected_adverse: A signature alone does not override failed supplier authorization or missing provenance.
observed_result: null
evidence_required:
- supplier inventory and risk assessment
- acquisition and notification agreements
- inspection/provenance records
- supplier offboarding and component-disposal evidence
```

Evidence and accountable completion. Cryptographic provenance records what was admitted and by whom. Supplier due diligence, counterfeit prevention, shipping inspections, contracting, and sustained external monitoring remain operational responsibilities with evidence owners.

Appendix H. Complete CSF 2.0 outcome case catalog

CSF 2.0 organizes outcomes into GOVERN, IDENTIFY, PROTECT, DETECT, RESPOND and RECOVER. This section includes all twenty-two categories and assigns a distinct original example to every one of the 106 Core outcome identifiers. Outcome numbers intentionally contain gaps because the official taxonomy contains those gaps. These cases are mappings and proposed workflows, not official NIST implementation examples or a claim of equivalence between frameworks. [N8]

An outcome can depend on several control families. For instance, managing identities combines enrollment, inventory, access policy, lifecycle and audit. The reverse also holds: one control family can support multiple functions. The raw cases therefore carry the outcome identifier and evidence needed for that outcome rather than assigning a universal “pass” based on a shared mechanism. The appropriate owner resolves scope and success criteria for the chosen organizational profile.

GOVERN examples assign accountable policy and risk decisions. IDENTIFY examples connect assets, information and changes to the actual boundary. PROTECT examples constrain execution and support the surrounding people and infrastructure safeguards. DETECT examples require observation and analysis. RESPOND examples connect approved containment to operational incident handling. RECOVER examples connect fresh authority to verified restoration and communication. The HTTP6 reference architecture supplies its strongest direct mechanisms at the authorization and containment boundaries; the other examples specify the integrations that make the organization-wide outcome meaningful.

H.1. GV.OC — Organizational Context

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.OC
fixture_status: illustrative_not_executed
owner: executive-risk-owner
observed_results: null
cases:
- outcome: GV.OC-01
  input: New reporting service proposed
  expected: Bind its root ceiling to the approved reporting mission
  evidence_required: Approved mission-to-system profile
- outcome: GV.OC-02
  input: Customers and finance operators need different records
  expected: Record each stakeholder expectation and affected data flow
  evidence_required: Stakeholder review and approved requirements
- outcome: GV.OC-03
  input: Contract adds a restricted-data processing obligation
  expected: Add the cited obligation to the selected requirement inventory
  evidence_required: Obligations register and review decision
- outcome: GV.OC-04
  input: Customers depend on timely reports
  expected: Attach delivery and recovery objectives to the service profile
  evidence_required: Service commitment and continuity analysis
- outcome: GV.OC-05
  input: Root service depends on hosting and key provider
  expected: Register external dependencies and responsible control providers
  evidence_required: Dependency inventory and provider evidence
```

H.2. GV.RM — Risk Management Strategy

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.RM
fixture_status: illustrative_not_executed
```

```
owner: enterprise-risk-owner
observed_results: null
cases:
- outcome: GV.RM-01
  input: Owners propose fleet automation
  expected: Agree risk objectives before root authority is approved
  evidence_required: Stakeholder-approved objective record
- outcome: GV.RM-02
  input: Owner proposes an offline lease
  expected: Compare the outage and revocation exposure with approved tolerance
  evidence_required: Risk-tolerance approval and lease rationale
- outcome: GV.RM-03
  input: Broker bypass finding is discovered
  expected: Escalate the finding into enterprise risk reporting
  evidence_required: Linked system and enterprise risk records
- outcome: GV.RM-04
  input: Threat exceeds tolerance
  expected: Select a documented treatment rather than silently broadening authority
  evidence_required: Treatment choice and authorized decision
- outcome: GV.RM-05
  input: Supplier reports key-service exposure
  expected: Deliver to named internal and supplier contacts
  evidence_required: Authenticated notification and acknowledgment
- outcome: GV.RM-06
  input: Two teams rate the same bypass differently
  expected: Apply the approved common risk-scoring method
  evidence_required: Risk method version and scored scenarios
- outcome: GV.RM-07
  input: New scoped-agent service could reduce manual exposure
  expected: Evaluate opportunity alongside new automation risks
  evidence_required: Opportunity and risk review
```

H.3. GV.RR — Roles, Responsibilities, and Authorities

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.RR
fixture_status: illustrative_not_executed
owner: executive-risk-owner
observed_results: null
cases:
- outcome: GV.RR-01
  input: High-impact root requested
  expected: Require the accountable organizational authority to approve scope
  evidence_required: Executive responsibility and approval record
- outcome: GV.RR-02
  input: Worker custody changes teams
  expected: Reassign owner through an approved admission transition
  evidence_required: Responsibility matrix and transfer evidence
- outcome: GV.RR-03
  input: Monitoring cannot meet review workload
  expected: Escalate staffing and service-capacity need
  evidence_required: Approved resourcing decision
- outcome: GV.RR-04
  input: Employee departure occurs
  expected: Connect personnel removal to affected roots and descendants
  evidence_required: Personnel event and access-removal review
```

H.4. GV.PO — Policy

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.PO
fixture_status: illustrative_not_executed
owner: policy-governance-owner
observed_results: null
cases:
```

```
- outcome: GV.PO-01
  input: New fleet policy is issued
  expected: Publish and approve policy meaning before activating its digest
  evidence_required: Approved and communicated policy version
- outcome: GV.PO-02
  input: Threat model changes allowed destinations
  expected: Issue a reviewed replacement rather than mutate live semantics
  evidence_required: Change impact and replacement admission
```

H.5. GV.OV — Oversight

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.OV
fixture_status: illustrative_not_executed
owner: oversight-board
observed_results: null
cases:
- outcome: GV.OV-01
  input: Quarterly evidence shows slow subtree revocation
  expected: Review the strategy and order corrective work
  evidence_required: Oversight decision and action owner
- outcome: GV.OV-02
  input: New regulation enters selected scope
  expected: Reconcile the obligations and control inventories
  evidence_required: Updated strategy and applicability record
- outcome: GV.OV-03
  input: Repeated bypass findings persist
  expected: Evaluate performance and approve improvements
  evidence_required: Trend report and management response
```

H.6. GV.SC — Cybersecurity Supply Chain Risk Management

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-GV.SC
fixture_status: illustrative_not_executed
owner: supplier-risk-owner
observed_results: null
cases:
- outcome: GV.SC-01
  input: New component supplier proposed
  expected: Use an approved supplier-risk program before admission
  evidence_required: Supplier program and approval
- outcome: GV.SC-02
  input: Hosted key provider and customer share duties
  expected: Name both sides of each inherited control dependency
  evidence_required: Shared responsibility agreement
- outcome: GV.SC-03
  input: Critical supplier becomes high risk
  expected: Update enterprise and system risk decisions together
  evidence_required: Linked supplier and enterprise risk entry
- outcome: GV.SC-04
  input: Dependency inventory adds a key-service vendor
  expected: Classify the dependency by criticality
  evidence_required: Supplier inventory and criticality rationale
- outcome: GV.SC-05
  input: Contract renewal is negotiated
  expected: Include artifact provenance and incident notification obligations
  evidence_required: Executed security requirements
- outcome: GV.SC-06
  input: Unreviewed vendor offers a signed runtime
  expected: Hold procurement pending due diligence
  evidence_required: Supplier review before agreement
- outcome: GV.SC-07
  input: Supplier vulnerability appears during service
  expected: Reassess admission and monitor remediation
```

```
  evidence_required: Supplier finding and treatment record
- outcome: GV.SC-08
  input: Provider outage exercise begins
  expected: Include the provider in response and recovery coordination
  evidence_required: Joint exercise record
- outcome: GV.SC-09
  input: Artifact changes during its support life
  expected: Monitor provenance and assurance obligations over its lifecycle
  evidence_required: Supplier lifecycle review
- outcome: GV.SC-10
  input: Vendor relationship ends
  expected: Revoke routes and admissions and verify data-return duties
  evidence_required: Offboarding and disposal evidence
```

H.7. ID.AM — Asset Management

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-ID.AM
fixture_status: illustrative_not_executed
owner: inventory-owner
observed_results: null
cases:
- outcome: ID.AM-01
  input: New machine requests admission
  expected: Register hardware identity and custody
  evidence_required: Hardware inventory record
- outcome: ID.AM-02
  input: New worker image is released
  expected: Register software version and service identity
  evidence_required: Build-to-runtime inventory
- outcome: ID.AM-03
  input: Worker creates a new permitted connection
  expected: Update authenticated data-flow representation
  evidence_required: Reviewed connection inventory
- outcome: ID.AM-04
  input: External audit service is introduced
  expected: Register service ownership and inherited duties
  evidence_required: Supplier-service inventory
- outcome: ID.AM-05
  input: Two assets compete for recovery priority
  expected: Use approved mission impact and classification
  evidence_required: Priority decision and impact record
- outcome: ID.AM-07
  input: Dataset adds contact information
  expected: Update data and metadata inventory
  evidence_required: Data inventory and classification
- outcome: ID.AM-08
  input: Host is retired
  expected: Retire identities and track disposal lifecycle
  evidence_required: Lifecycle and retirement records
```

H.8. ID.RA — Risk Assessment

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-ID.RA
fixture_status: illustrative_not_executed
owner: risk-assessment-owner
observed_results: null
cases:
- outcome: ID.RA-01
  input: Adapter scanner reports an input defect
  expected: Validate and register the vulnerable path
  evidence_required: Validated vulnerability finding
- outcome: ID.RA-02
  input: Threat advisory describes identity relay
```

```
  expected: Ingest and assess relevance to admitted protocols
  evidence_required: Threat-intelligence review
- outcome: ID.RA-03
  input: Insider proposes broad root permissions
  expected: Record internal abuse scenario
  evidence_required: Threat scenario and affected assets
- outcome: ID.RA-04
  input: Root outage could delay critical processing
  expected: Estimate impact and likelihood with owners
  evidence_required: Scenario analysis
- outcome: ID.RA-05
  input: Bypass route and stale role cache coexist
  expected: Prioritize their combined exposure
  evidence_required: Risk prioritization record
- outcome: ID.RA-06
  input: Remediation options are available
  expected: Choose and track accountable treatment
  evidence_required: Treatment plan and closure evidence
- outcome: ID.RA-07
  input: Exception requests a direct network route
  expected: Assess and approve or reject with expiry
  evidence_required: Exception decision and reassessment
- outcome: ID.RA-08
  input: Researcher reports nonce-reuse behavior
  expected: Receive and validate disclosure through a managed channel
  evidence_required: Disclosure and remediation tracking
- outcome: ID.RA-09
  input: Signed module arrives from distributor
  expected: Check provenance and integrity before admission
  evidence_required: Acquisition integrity inspection
- outcome: ID.RA-10
  input: Candidate key-service supplier is critical
  expected: Assess supplier before commitment
  evidence_required: Pre-acquisition supplier assessment
```

H.9. ID.IM — Improvement

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-ID.IM
fixture_status: illustrative_not_executed
owner: continuous-improvement-owner
observed_results: null
cases:
- outcome: ID.IM-01
  input: Assessment discovers an uncovered resource
  expected: Revise the gate architecture and scope inventory
  evidence_required: Evaluation-driven improvement record
- outcome: ID.IM-02
  input: Joint recovery drill exceeds objective
  expected: Correct the drill findings with provider participation
  evidence_required: Exercise findings and retest
- outcome: ID.IM-03
  input: Repeated production denial reveals a policy error
  expected: Review and replace policy through controlled admission
  evidence_required: Operational lesson and change approval
- outcome: ID.IM-04
  input: Recovery runbook no longer matches deployment
  expected: Update communicate and exercise the plan
  evidence_required: Maintained plan and exercise record
```

H.10. PR.AA — Identity Management, Authentication, and Access Control

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-PR.AA
fixture_status: illustrative_not_executed
```

```
owner: identity-and-access-owner
observed_results: null
cases:
- outcome: PR.AA-01
  input: Child reaches end of service
  expected: Retire its identity key and renewal authority
  evidence_required: Lifecycle and key-retirement records
- outcome: PR.AA-02
  input: Person enrolls through the Human Admission Client
  expected: Bind account to the selected completed proofing process
  evidence_required: Proofing and authenticator-binding evidence
- outcome: PR.AA-03
  input: Machine opens a twin session
  expected: Verify current instance possession and admission
  evidence_required: Authenticated handshake trace
- outcome: PR.AA-04
  input: Assertion is replayed at a different service
  expected: Reject wrong audience and consumed challenge
  evidence_required: Assertion-substitution test
- outcome: PR.AA-05
  input: Child asks for a broader role
  expected: Apply the ancestor ceiling and separation-of-duty rules
  evidence_required: Policy decision and approval tests
- outcome: PR.AA-06
  input: Visitor requests access to root-host room
  expected: Apply facility authorization and monitoring
  evidence_required: Physical access and escort evidence
```

H.11. PR.AT — Awareness and Training

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-PR.AT
fixture_status: illustrative_not_executed
owner: training-program-owner
observed_results: null
cases:
- outcome: PR.AT-01
  input: New operator joins reporting team
  expected: Deliver general security training and verify completion
  evidence_required: Curriculum delivery and comprehension record
- outcome: PR.AT-02
  input: Operator becomes root approver
  expected: Require role-specific delegation and recovery exercise
  evidence_required: Specialized training evidence
```

H.12. PR.DS — Data Security

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-PR.DS
fixture_status: illustrative_not_executed
owner: data-protection-owner
observed_results: null
cases:
- outcome: PR.DS-01
  input: Worker requests encrypted ledger data
  expected: Gate key use and maintain protected available storage
  evidence_required: Key-release and storage resilience evidence
- outcome: PR.DS-02
  input: Report crosses a network
  expected: Authenticate peers and protect payload and transport
  evidence_required: Channel and integrity tests
- outcome: PR.DS-10
  input: Sensitive record is processed in memory
  expected: Use the admitted isolation and output-minimization path
  evidence_required: Runtime isolation and processing tests
```

```
- outcome: PR.DS-11
  input: Backup job completes
  expected: Protect and restore-test the resulting backup
  evidence_required: Backup integrity and restore record
```

H.13. PR.PS — Platform Security

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-PR.PS
fixture_status: illustrative_not_executed
owner: platform-owner
observed_results: null
cases:
- outcome: PR.PS-01
  input: Normalizer configuration is changed
  expected: Require measured controlled replacement
  evidence_required: Baseline and change-control record
- outcome: PR.PS-02
  input: Unsupported dependency is detected
  expected: Replace or remove through an approved release
  evidence_required: Patch and support lifecycle evidence
- outcome: PR.PS-03
  input: Hardware reaches service limit
  expected: Replace and revoke its instance registrations
  evidence_required: Hardware maintenance and disposal record
- outcome: PR.PS-04
  input: Worker executes an approved action
  expected: Collect protected decision and outcome records
  evidence_required: Audit generation and collection test
- outcome: PR.PS-05
  input: Application attempts arbitrary child execution
  expected: Deny unapproved executable admission
  evidence_required: Execution-control negative test
- outcome: PR.PS-06
  input: Policy engine is developed
  expected: Apply secure build review testing and vulnerability handling
  evidence_required: Development assurance package
```

H.14. PR.IR — Technology Infrastructure Resilience

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-PR.IR
fixture_status: illustrative_not_executed
owner: resilience-owner
observed_results: null
cases:
- outcome: PR.IR-01
  input: Worker opens an unbrokered socket
  expected: Block unauthorized logical access
  evidence_required: Route and broker-bypass test
- outcome: PR.IR-02
  input: Facility cooling alarm is raised
  expected: Invoke the approved environmental response
  evidence_required: Facility exercise and system response
- outcome: PR.IR-03
  input: Primary authority service fails
  expected: Use authorized recovery with consistent epoch state
  evidence_required: Failover and restoration test
- outcome: PR.IR-04
  input: Thousands of legitimate workers request proofs
  expected: Maintain capacity without bypassing policy
  evidence_required: Load and resource-exhaustion evidence
```

H.15. DE.CM — Continuous Monitoring

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-DE.CM
fixture_status: illustrative_not_executed
owner: monitoring-owner
observed_results: null
cases:
- outcome: DE.CM-01
  input: Unexpected destination traffic appears
  expected: Detect compare against route policy and alert
  evidence_required: Network monitoring observation
- outcome: DE.CM-02
  input: Host enclosure alarm occurs
  expected: Correlate physical event with admitted machines
  evidence_required: Facility monitoring record
- outcome: DE.CM-03
  input: Administrator repeatedly requests broader scopes
  expected: Review activity under approved monitoring purpose
  evidence_required: Personnel-activity review
- outcome: DE.CM-06
  input: External provider changes runtime unexpectedly
  expected: Detect provider assurance drift
  evidence_required: Provider monitoring finding
- outcome: DE.CM-09
  input: Child measurement diverges
  expected: Detect runtime inconsistency and contain authority
  evidence_required: Runtime monitoring event
```

H.16. DE.AE — Adverse Event Analysis

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-DE.AE
fixture_status: illustrative_not_executed
owner: security-analysis-owner
observed_results: null
cases:
- outcome: DE.AE-02
  input: Repeated invalid proof positions appear
  expected: Analyze whether faults indicate replay or malfunction
  evidence_required: Analyst decision record
- outcome: DE.AE-03
  input: Door alarm coincides with cloned-instance attempts
  expected: Correlate facility identity and network events
  evidence_required: Multi-source incident analysis
- outcome: DE.AE-04
  input: Scheduler key-use anomaly is confirmed
  expected: Enumerate affected descendants and resources
  evidence_required: Validated scope estimate
- outcome: DE.AE-06
  input: Analysis identifies a high-risk subtree
  expected: Deliver restricted evidence to authorized responders
  evidence_required: Analyst handoff and receipt
- outcome: DE.AE-07
  input: Threat intelligence names a new adapter attack
  expected: Enrich the event analysis with relevant context
  evidence_required: Context-linked analysis
- outcome: DE.AE-08
  input: Events meet approved declaration criteria
  expected: Declare incident through the incident workflow
  evidence_required: Incident declaration record
```

H.17. RS.MA — Incident Management

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RS.MA
fixture_status: illustrative_not_executed
owner: incident-commander
observed_results: null
cases:
- outcome: RS.MA-01
  input: Incident is declared
  expected: Execute the coordinated response plan
  evidence_required: Incident execution record
- outcome: RS.MA-02
  input: Automated alert opens a case
  expected: Validate the report before assigning conclusions
  evidence_required: Triage finding
- outcome: RS.MA-03
  input: Critical root and low-impact worker alerts compete
  expected: Prioritize by documented impact
  evidence_required: Severity and priority record
- outcome: RS.MA-04
  input: Response exceeds team authority
  expected: Escalate to the designated decision maker
  evidence_required: Escalation acknowledgment
- outcome: RS.MA-05
  input: Eradication reaches recovery criteria
  expected: Authorize recovery through the approved gate
  evidence_required: Recovery initiation decision
```

H.18. RS.AN — Incident Analysis

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RS.AN
fixture_status: illustrative_not_executed
owner: forensics-owner
observed_results: null
cases:
- outcome: RS.AN-03
  input: Unauthorized effect is reported
  expected: Reconstruct cause from policy and transaction evidence
  evidence_required: Root-cause analysis
- outcome: RS.AN-06
  input: Investigator exports audit evidence
  expected: Record analyst actions and preserve provenance
  evidence_required: Investigation activity trail
- outcome: RS.AN-07
  input: Runtime snapshot is collected
  expected: Preserve evidence and metadata with controlled custody
  evidence_required: Evidence collection receipt
- outcome: RS.AN-08
  input: Affected subtree count is initially estimated
  expected: Validate magnitude against registry and resource outcomes
  evidence_required: Confirmed incident scope
```

H.19. RS.CO — Incident Response Reporting and Communication

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RS.CO
fixture_status: illustrative_not_executed
owner: incident-communications-owner
observed_results: null
cases:
- outcome: RS.CO-02
```

```
  input: Incident triggers reporting obligations
  expected: Notify the approved parties through reviewed channels
  evidence_required: Notification and delivery records
- outcome: RS.CO-03
  input: Partner needs investigation information
  expected: Share only authorized case data
  evidence_required: Approved sharing and receipt record
```

H.20. RS.MI — Incident Mitigation

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RS.MI
fixture_status: illustrative_not_executed
owner: incident-commander
observed_results: null
cases:
- outcome: RS.MI-01
  input: Worker compromise is confirmed
  expected: Fence the affected lineage at enforcing resources
  evidence_required: Containment and last-accepted-action evidence
- outcome: RS.MI-02
  input: Root cause is a vulnerable adapter
  expected: Remove the malicious or defective component before restart
  evidence_required: Eradication and clean-admission evidence
```

H.21. RC.RP — Incident Recovery Plan Execution

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RC.RP
fixture_status: illustrative_not_executed
owner: recovery-owner
observed_results: null
cases:
- outcome: RC.RP-01
  input: Recovery is authorized
  expected: Execute the selected runbook under a new admitted epoch
  evidence_required: Recovery execution record
- outcome: RC.RP-02
  input: Many services need restoration
  expected: Prioritize within approved resources and mission needs
  evidence_required: Ordered recovery plan
- outcome: RC.RP-03
  input: Backup is selected for use
  expected: Verify integrity and compromise checks before restoration
  evidence_required: Backup verification evidence
- outcome: RC.RP-04
  input: Critical reporting must resume first
  expected: Use mission objectives to set post-incident service conditions
  evidence_required: Business-owner recovery decision
- outcome: RC.RP-05
  input: Restored child reconnects
  expected: Check integrity service behavior and intended authority
  evidence_required: Restoration and functional validation
- outcome: RC.RP-06
  input: Recovery objectives are met
  expected: Declare completion and finalize records
  evidence_required: Recovery closure approval
```

H.22. RC.CO — Incident Recovery Communication

The following cases map this category to proposed architecture operation and its required organizational evidence. [N8]

```
example_group: CSF-RC.CO
fixture_status: illustrative_not_executed
owner: recovery-communications-owner
observed_results: null
cases:
- outcome: RC.CO-03
  input: Approved restoration milestone is reached
  expected: Communicate scoped progress to designated stakeholders
  evidence_required: Progress notice and delivery record
- outcome: RC.CO-04
  input: Public service status changes
  expected: Publish reviewed recovery messaging through approved means
  evidence_required: Public communication approval
```

Appendix I. Assurance, categorization, CUI, and lifecycle cases

I.1. Security categorization and minimum security areas

I.1.1. FIPS 199: separate confidentiality, integrity and availability

FIPS 199 supplies security categorization based on consequences to confidentiality, integrity and availability. The organization determines impact for its actual mission and information; the architecture does not assign a category merely because it uses a certain key size or admission mechanism. In the examples below, low, moderate and high are illustrative impact selections whose appropriateness must be established by the owner. [E1]

The HTTP6 reference architecture answers confidentiality concerns through controlled disclosure, scoped key release and protected communication. It answers integrity concerns through exact-action binding, authenticated policy and execution state, and verification at the effect boundary. Availability requires capacity, independent infrastructure, service continuity and recovery in addition to authorization. A cryptographic fail-closed gate preserves an access rule during uncertainty; the continuity design establishes when authorized work can resume.

```
fixture_status: illustrative_not_executed
cases:
- objective: confidentiality
  illustrative_impact: low
  input: An authorized public-data service accidentally reveals a minor internal status label.
  expected_architecture_response: Use purpose- and recipient-bound resource gates, protected transport,
    controlled decryption and data-flow restrictions.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: confidentiality
  illustrative_impact: moderate
  input: A reporting service exposes a consequential set of customer records.
  expected_architecture_response: Use purpose- and recipient-bound resource gates, protected transport,
    controlled decryption and data-flow restrictions.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: confidentiality
  illustrative_impact: high
  input: A critical service exposes information whose loss has severe or catastrophic consequences in
    the selected mission.
  expected_architecture_response: Use purpose- and recipient-bound resource gates, protected transport,
    controlled decryption and data-flow restrictions.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: integrity
  illustrative_impact: low
  input: A noncritical status field is changed without approval.
  expected_architecture_response: Bind the canonical operation and payload to fresh proof, protect
  ↪  manifests
    and state, and validate the durable effect.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: integrity
  illustrative_impact: moderate
  input: A financial reporting dataset is altered and causes significant business harm.
  expected_architecture_response: Bind the canonical operation and payload to fresh proof, protect
  ↪  manifests
    and state, and validate the durable effect.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: integrity
```

```
  illustrative_impact: high
  input: Unauthorized changes affect critical commands or decisions with severe or catastrophic
  ↪  consequences.
  expected_architecture_response: Bind the canonical operation and payload to fresh proof, protect
  ↪  manifests
    and state, and validate the durable effect.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: availability
  illustrative_impact: low
  input: A noncritical informational service suffers a limited interruption.
  expected_architecture_response: Provide assessed redundancy, capacity, recovery staffing and
  ↪  bounded-failure
    procedures while preserving authorization gates.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: availability
  illustrative_impact: moderate
  input: An operational reporting service becomes unavailable with serious mission consequences.
  expected_architecture_response: Provide assessed redundancy, capacity, recovery staffing and
  ↪  bounded-failure
    procedures while preserving authorization gates.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
- objective: availability
  illustrative_impact: high
  input: A critical service outage creates severe or catastrophic consequences in the selected mission.
  expected_architecture_response: Provide assessed redundancy, capacity, recovery staffing and
  ↪  bounded-failure
    procedures while preserving authorization gates.
  evidence_required: Approved impact analysis for the actual information and mission, selected baseline
    and measured safeguards
  observed_result: null
```

A system's selected control baseline follows the applicable categorization and tailoring procedure, including the required treatment of its information types. The mapping retains the approved reasoning and the organization-defined choices. None of the example impact assignments is a recommendation for categorizing an unspecified real deployment. [N16; E1, E2]

I.1.2. FIPS 200: all seventeen minimum security areas

FIPS 200 defines minimum security areas within its applicability. The table below gives an explicit route to a raw example for each area. The three additional SP 800-53 families PM, PT and SR receive their own examples in Appendix G rather than being silently forced into a seventeen-area list. The minimum-area names and the modern catalog-family names are related organizational classifications; fulfilling an area requires its actual selected safeguards. [E2; N1]

FIPS 200 area	Raw example in this edition	Concrete evidence path
Access Control	EX-AC-01	Approved requests succeed; prohibited export fails at the resource
Awareness and Training	EX-AT-01	Actual instruction and verified eligibility records
Audit and Accountability	EX-AU-01	Protected human-to-action trace and independent record review

FIPS 200 area	Raw example in this edition	Concrete evidence path
Certification, Accreditation, and Security Assessments	EX-CA-01	Actual assessment and the applicable authorization process
Configuration Management	EX-CM-01	Measured configuration and controlled replacement evidence
Contingency Planning	EX-CP-01	Tested recovery preserving consumed state and fresh generations
Identification and Authentication	EX-IA-01	Enrollment, factor, context, replay and lifecycle evidence
Incident Response	EX-IR-01	Triage, authorized containment, reporting and recovery records
Maintenance	EX-MA-01	Approved ticket, technician, tools and terminated privileges
Media Protection	EX-MP-01	Custody, selected sanitization and actual verification
Physical and Environmental Protection	EX-PE-01	Facility safeguards and assessed hosting dependencies
Planning	EX-PL-01	Current boundary and security planning with every usable route assigned
Personnel Security	EX-PS-01	Personnel decisions and timely privilege removal
Risk Assessment	EX-RA-01	Threat, impact, vulnerability and treatment evidence
System and Services Acquisition	EX-SA-01	Security requirements, supplier deliverables and acceptance review
System and Communications Protection	EX-SC-01	Scoped communication and protected security functions
System and Information Integrity	EX-SI-01	Integrity, vulnerability, malware and input-validation evidence

I.2. Identity assurance: raw cases for proofing, authentication and federation

I.2.1. Preserve the three independent assurance axes

The SP 800-63-4 suite treats identity proofing, authentication and federation as separate matters. An IAL concerns the identity proofing and enrollment process, an AAL concerns the authentication transaction, and an FAL concerns federation. A machine's valid ancestry does not automatically supply a person's proofing evidence or a new human-authentication event. [N3–N5]

In the source architecture, Rosario–Wang private-reference verification supplies an enrollment-related admission mechanism; protected device keys and action-bound messages supply separate cryptographic functions. The deployment selects a supported authenticator profile, characterizes the custom relation and integrates the remaining applicable requirements. A private seed's bit length, the number of rings, and a diagram labeled "zero knowledge" are not substitutes for that complete profile. The positive engineering statement is that each assurance property has a named mechanism and a complete evidence path. [S1, II.2–II.7]

The cases below distinguish a service that requires no identity proofing from IAL1. Under the selected Revision 4 framework, IAL1 has proofing requirements; it is not shorthand for an entirely unverified identity. Similarly, a FAL3 relying-party authenticator relationship is evaluated for the subscriber and federation transaction rather than inferred from any descendant workload's signing key. [N4, N5]

No proofing selected

```
group: Identity proofing
classification: No proofing selected
fixture_status: illustrative_not_executed
input: A service needs only a pseudonymous subscriber relationship.
expected: Bind the account and authenticator without asserting a civil identity or IAL claim.
evidence_required: Documented service risk analysis and selected proofing decision
observed_result: null
```

IAL1

```
group: Identity proofing
classification: IAL1
fixture_status: illustrative_not_executed
input: A person enrolls for a service whose selected profile requires IAL1.
expected: The proofing provider performs the required identity resolution, evidence and attribute
↪  validation,
  and applicant verification; the Enrollment Service binds the resulting enrollment receipt to the account.
evidence_required: Complete IAL1 procedure, provider evidence, fraud controls, enrollment binding,
↪  notifications
  and redress records
observed_result: null
```

IAL2

```
group: Identity proofing
classification: IAL2
fixture_status: illustrative_not_executed
input: A person requests access to a more consequential service.
expected: Require the selected IAL2 proofing procedure before issuing its identity-bound root; possession
  of the private reference alone does not supply the proofing evidence.
evidence_required: All applicable IAL2 requirements and assessment records for the chosen proofing pathway
observed_result: null
```

IAL3

```
group: Identity proofing
classification: IAL3
fixture_status: illustrative_not_executed
input: The selected service requires high-confidence person-to-identity binding.
expected: Integrate the required on-site attended process, trained proofing personnel and biometric
↪  requirements
  before the Enrollment Service binds the enrollment.
evidence_required: Complete IAL3 procedure and protected proofing evidence, with enrollment and privacy
  review
observed_result: null
```

AAL1

```
group: Authentication assurance
classification: AAL1
fixture_status: illustrative_not_executed
input: A low-impact service accepts a selected single-factor authentication method.
expected: Admit a scoped root only after the permitted account-bound authenticator succeeds; descendants
  remain machine identities.
evidence_required: Applicable AAL1 authenticator, binding, replay, session and lifecycle evidence
observed_result: null
```

AAL2

```
group: Authentication assurance
classification: AAL2
fixture_status: illustrative_not_executed
input: A service requires two distinct authentication factors.
expected: Bind two distinct factors in the selected authenticator configuration and provide the required
  phishing-resistant option; multiple regions of one remembered secret are one factor.
evidence_required: Complete AAL2 transaction evidence, factor integration, intent, rate limits, recovery
  and session tests
observed_result: null
```

AAL3

```
group: Authentication assurance
classification: AAL3
fixture_status: illustrative_not_executed
input: A high-assurance root is requested.
expected: Use the required public-key cryptographic authenticator with non-exportable key, phishing
↪  resistance,
  two distinct factors and all remaining applicable AAL3 requirements.
evidence_required: Actual authenticator and verifier profile, custody and applicable module validation
  evidence, complete AAL3 assessment
observed_result: null
```

FAL1

```
group: Federation assurance
classification: FAL1
fixture_status: illustrative_not_executed
input: A separately administered relying party receives a subscriber assertion.
expected: Validate the issuer, assertion protection, audience, freshness and other FAL1 requirements
↪  before
  using its identity claims in a root admission.
evidence_required: Trust configuration, assertion validation records and complete FAL1 handling tests
observed_result: null
```

FAL2

```
group: Federation assurance
classification: FAL2
fixture_status: illustrative_not_executed
input: A relying-party initiated federation transaction is redirected through an adversarial browser.
expected: Implement the selected FAL2 protections for assertion injection and the required transaction
  and relying-party binding; retain the separate architecture action proof.
evidence_required: Complete FAL2 trust and transaction evidence including injected and wrong-audience
  assertion tests
observed_result: null
```

FAL3

```
group: Federation assurance
classification: FAL3
fixture_status: illustrative_not_executed
input: A high-assurance relying party needs proof beyond possession of an assertion.
expected: Implement the required subscriber authenticator proof at the relying party, such as the specified
  holder-of-key or bound-authenticator approach, with the remaining FAL3 requirements.
evidence_required: Complete FAL3 evidence connecting the subscriber, assertion, relying-party
↪  authenticator
  and trusted party configuration
observed_result: null
```

I.2.2. Complete the mandatory details, including interaction and recovery

Each assurance profile expands into its complete applicable normative clauses. The register records selected authenticator type, factor independence, key protection, rate limits, channel and verifier binding, intent, binding and replacement procedures, account recovery, subscriber notification, privacy, redress and session conditions. Accessibility and usability decisions also retain their applicable requirements and evidence. A gesture sequence that discloses only membership witnesses can improve an interaction model while still requiring accessible presentation and secure recovery. [N3–N5]

For automation, record separate human_auth_time, root_generation, workload_generation, channel_epoch and last_machine_activity. Heartbeats can establish machine liveness; they do not refresh the human's authentication time. Where a human session expires, the system requests the required human reauthentication even when permitted noninteractive work has its own separately authorized lifetime. Policy declares which activities require current human presence and which are deliberately delegated.

```
example_id: IDENTITY-CONTINUITY-01
fixture_status: illustrative_not_executed
human_auth_time: '2026-11-09T17:00:00Z'
last_machine_activity: '2026-11-09T20:00:00Z'
root_generation: root-e42
workload_generation: report-worker-17
request: protected-human-session-renewal
required_evidence: selected-profile-human-reauthentication
machine_heartbeat_supplied: true
fresh_human_ceremony_supplied: false
expected: reject-human-session-renewal-and-follow-approved-reauthentication-path
observed: null
```

Enrollment and recovery use the same separation. A historical secret may contribute only the evidence authorized by the selected recovery procedure. A successful recovery establishes the permitted new account or authenticator state, issues required notifications, and fences superseded authority. Sensitive changes are not approved simply because a person can reconstruct an old geometric relationship. The authoritative enrollment and recovery service performs all requirements selected for that operation. [N3, N4; S1, II.7.3]

I.3. Cryptographic module assurance and private material

I.3.1. All eleven FIPS 140-3 requirement areas

The module boundary is explicitly distinguished from the complete HTTP6 reference architecture. FIPS 140-3 identifies eleven module-security areas and four increasing qualitative levels. The examples below associate each area with the actual cryptographic component and its applicable evidence. They do not convert a custom authentication relation or a complete operating system into a validated module by association. [N13, N14]

I.3.1.1. Cryptographic module specification

```
area: Cryptographic module specification
fixture_status: illustrative_not_executed
input: A root service uses a hardware security device and an external policy broker.
expected: Define which components are inside the validated module boundary and which architecture protocol
↪  components
  are outside it.
evidence_required: Applicable module certificate, security policy, exact version and boundary diagram
observed_result: null
```

I.3.1.2. Cryptographic module interfaces

```
area: Cryptographic module interfaces
fixture_status: illustrative_not_executed
input: An application requests raw key export through a diagnostic API.
expected: Expose only admitted services through the selected module interface; separately enforce the
  Delegation Broker policy.
evidence_required: Module interface documentation and configured-service tests
observed_result: null
```

I.3.1.3. Roles, services, and authentication

```
area: Roles, services, and authentication
fixture_status: illustrative_not_executed
input: A worker attempts a cryptographic-officer operation.
expected: Apply module-specific role authentication and service authorization, in addition to the worker
  lineage checks.
evidence_required: Module role/service matrix and role-change tests
observed_result: null
```

I.3.1.4. Software/firmware security

```
area: Software/firmware security
fixture_status: illustrative_not_executed
input: A firmware update changes the cryptographic service.
expected: Verify approved firmware integrity and its relationship to the certified configuration before
  admitting the new generation.
evidence_required: Signed firmware provenance, applicable validation configuration and update evidence
observed_result: null
```

I.3.1.5. Operating environment

```
area: Operating environment
fixture_status: illustrative_not_executed
input: A binary is moved to an unlisted operating environment.
expected: Check the applicable validation and deployment restrictions before using the module in the
↪  selected
  mode.
evidence_required: Operational environment inventory and validation applicability review
observed_result: null
```

I.3.1.6. Physical security

```
area: Physical security
fixture_status: illustrative_not_executed
input: A host enclosure is opened.
expected: Use the physical safeguards required by the selected module profile and separately enforce the
  facility response plan.
evidence_required: Module physical-security evidence and installed-environment inspection
observed_result: null
```

I.3.1.7. Non-invasive security

```
area: Non-invasive security
fixture_status: illustrative_not_executed
input: An attacker records power or electromagnetic observations.
expected: Evaluate the applicable non-invasive requirements and selected implementation protections;
↪  ordinary
  transcript privacy is a different property.
evidence_required: Applicable laboratory evidence and deployment-side leakage analysis
observed_result: null
```

I.3.1.8. Sensitive security parameter management

```
area: Sensitive security parameter management
fixture_status: illustrative_not_executed
input: A retired child retains an obsolete session secret.
expected: Use the module-approved generation, storage, access and destruction services with Authoritative
↪  State Plane generation
  retirement.
evidence_required: Key lifecycle records, zeroization procedure and applicable service tests
observed_result: null
```

I.3.1.9. Self-tests

```
area: Self-tests
fixture_status: illustrative_not_executed
input: A required module self-test fails before a signing operation.
expected: Use the specified module error state and withhold the protected cryptographic service; the broker
  does not fabricate an authorization.
evidence_required: Self-test and error-state evidence for the actual module configuration
observed_result: null
```

I.3.1.10. Life-cycle assurance

```
area: Life-cycle assurance
fixture_status: illustrative_not_executed
input: A released module is replaced by an untracked build.
expected: Require supply, build, installation and maintenance provenance for the applicable validated
  version.
evidence_required: Lifecycle documentation, release provenance and installation review
observed_result: null
```

I.3.1.11. Mitigation of other attacks

```
area: Mitigation of other attacks
fixture_status: illustrative_not_executed
input: The module claims a specific additional attack mitigation.
expected: Record its stated scope, configuration and required evidence; connect it to the deployment threat
  model.
evidence_required: Security-policy claim and corresponding validation or technical evidence
observed_result: null
```

I.3.2. Four level-selection examples

The required level is selected under the applicable program and risk profile. The actual certificate, security policy, operational environment and permitted services determine whether the chosen module meets that selection. A system may use several modules for different purposes. Each relationship is recorded separately, including any relevant per-area qualifications and operational restrictions. [N13, N14]

```
group: Cryptographic module level
classification: FIPS 140-3 Level 1
fixture_status: illustrative_not_executed
input: A deployment profile selects Level 1.
expected: Use an applicable validated module meeting the selected Level 1 requirements and configured
  approved services; verify certificate scope and deployment conditions. Do not infer the level from a
```

```
  TEE name, key length, or admission ceremony geometry.
evidence_required: Actual module certificate, per-area qualifications where relevant, security policy
  and operational-environment evidence
observed_result: null
```

```
group: Cryptographic module level
classification: FIPS 140-3 Level 2
fixture_status: illustrative_not_executed
input: A deployment profile selects Level 2.
expected: Use an applicable validated module meeting the selected Level 2 requirements and configured
  approved services; verify certificate scope and deployment conditions. Do not infer the level from a
  TEE name, key length, or admission ceremony geometry.
evidence_required: Actual module certificate, per-area qualifications where relevant, security policy
  and operational-environment evidence
observed_result: null
```

```
group: Cryptographic module level
classification: FIPS 140-3 Level 3
fixture_status: illustrative_not_executed
input: A deployment profile selects Level 3.
expected: Use an applicable validated module meeting the selected Level 3 requirements and configured
  approved services; verify certificate scope and deployment conditions. Do not infer the level from a
  TEE name, key length, or admission ceremony geometry.
evidence_required: Actual module certificate, per-area qualifications where relevant, security policy
  and operational-environment evidence
observed_result: null
```

```
group: Cryptographic module level
classification: FIPS 140-3 Level 4
fixture_status: illustrative_not_executed
input: A deployment profile selects Level 4.
expected: Use an applicable validated module meeting the selected Level 4 requirements and configured
  approved services; verify certificate scope and deployment conditions. Do not infer the level from a
  TEE name, key length, or admission ceremony geometry.
evidence_required: Actual module certificate, per-area qualifications where relevant, security policy
  and operational-environment evidence
observed_result: null
```

I.3.3. Entropy, fresh randomness and approved service use

The private binary is recorded as input material with provenance, entropy assessment, storage and access rules. A timestamp indexes a construction; it contributes no presumed secret entropy. The computational profile uses protected keying material and specified derivation. The exact private-mask profile allocates genuinely fresh private randomness under its stated model and tracks use. Each purpose receives domain-separated material. [S1, II.2–II.4; N10–N12]

```
example_id: CRYPTO-PROFILE-01
fixture_status: illustrative_not_executed
profile: computational-geometry-and-protected-action-authentication
entropy_source_assessment: null
rbg_construction_reference: null
module:
  certificate_identifier: null
  module_version: null
  operational_environment: null
  approved_service_configuration: null
key_purposes: [geometry, channel, issuance, audit]
domain_separation_required: true
private_mask_profile:
  selected: false
  fresh_randomness_budget: not_applicable_to_this_selected_profile
request: declare-profile-production-ready
expected: hold-until-required-cryptographic-and-implementation-evidence-is-accepted
observed: null
```

The module service used for each operation is part of the evidence: encryption, authenticated encryption, signatures, message authentication, derivation, key establishment or random-bit generation. The selected NIST recommendations and validation requirements govern those services

where applicable. Crypto-agility changes are admitted as versioned replacements and trigger impact assessment; changing the algorithm name in an application configuration does not update the proof of deployment equivalence. The relevant target is protected key use throughout the admitted lifetime, not a large nominal number in a diagram.

I.4. CUI requirements: all seventeen Revision 3 families and selected enhancements

SP 800-171 Revision 3 addresses CUI protection in nonfederal systems within its stated scope. Its seventeen requirement families use their own identifiers and requirement text. They are not interchangeable with the similarly named SP 800-53 families. This section supplies a concrete example for each CUI family while preserving its namespace; the selected requirements and assessment objectives must be imported from SP 800-171 and SP 800-171A. [E3]

The governing agreement identifies the applicable edition, system boundary, selected parameters and obligations. Publication of a newer edition does not by itself rewrite a particular agreement. Within that declared scope, the Policy Decision Point binds CUI actions to an admitted human origin, workload identity, purpose, destination, authorized operation and current state. The hosting, personnel, maintenance, incident and supplier integrations below supply the complementary operational safeguards.

03.01 — Access Control

```
family: '03.01'
name: Access Control
fixture_status: illustrative_not_executed
source_example: EX-AC-01
input: A reporting worker has legitimate ledger access but is instructed to export the complete customer
  table to an unrelated host. A permissive intermediary could otherwise turn its own broad service account
  into an escape from the worker's restrictions.
expected_system_response: An approved aggregate reaches the report service through the same gate and
↪  consumes
  one action position.
expected_adverse: The raw export is denied at both the egress broker and destination-facing resource
↪  adapter;
  no broader broker identity replaces the caller’s scope.
evidence_required:
- approved role assignments
- ancestor policies and canonical digests
- positive and negative request traces
- independent resource-side effect inspection
observed_result: null
```

03.02 — Awareness and Training

```
family: '03.02'
name: Awareness and Training
fixture_status: illustrative_not_executed
source_example: EX-AT-01
input: An administrator can complete a cryptographic ceremony while lacking the role-specific knowledge
  needed to approve a fleet safely. Authentication alone does not teach incident handling or safe
  ↪  delegation.
expected_system_response: A trained administrator with a current verified record can initiate an otherwise
  approved root ceremony.
expected_adverse: A missing, forged, or expired training record holds administrative admission and directs
  the person to the approved training process.
evidence_required:
- approved curriculum
- delivery and completion records
- role-specific exercise results
- training-record issuer validation tests
observed_result: null
```

03.03 — Audit and Accountability

```
family: '03.03'
name: Audit and Accountability
fixture_status: illustrative_not_executed
source_example: EX-AU-01
input: A valid action leaves a log that identifies only a shared service account, or the worker can erase
  the evidence of its own prohibited request.
expected_system_response: Authorized reviewers reconstruct an approved report from birth through effect
  and verify record provenance.
expected_adverse: The worker's deletion request fails, and deliberate collection interruption raises the
  configured audit-failure response.
evidence_required:
- event-selection policy
- protected sample records
- collector outage exercise
- retention and reviewer-access configuration
observed_result: null
```

03.04 — Configuration Management

```
family: '03.04'
name: Configuration Management
fixture_status: illustrative_not_executed
source_example: EX-CM-01
input: An unchanged child policy name silently picks up a broader role definition, or a patched normalizer
  interprets a formerly denied destination as allowed.
expected_system_response: An approved replacement with matching measurements receives a new admission
  under the existing root ceiling.
expected_adverse: The altered role catalog fails the admitted commitment and cannot expand the existing
  child's authority.
evidence_required:
- approved baseline manifest
- change request and approvers
- artifact digest comparison
- rollout and rollback test results
observed_result: null
```

03.05 — Identification and Authentication

```
family: '03.05'
name: Identification and Authentication
fixture_status: illustrative_not_executed
source_example: EX-IA-01
input: A recorded human response is replayed at another verifier, or a workload identity is confused with
  a fresh interaction by the human who originally approved it.
expected_system_response: A fresh ceremony bound to the genuine verifier activates only the named instance
  and policy.
expected_adverse: The old nonce or different audience fails; machine heartbeats never change
↪  human_auth_time.
evidence_required:
- enrollment and binding records
- authenticator profile and key-custody evidence
- cross-origin and replay tests
- human timeout and recovery tests
observed_result: null
```

03.06 — Incident Response

```
family: '03.06'
name: Incident Response
fixture_status: illustrative_not_executed
source_example: EX-IR-01
input: One compromised worker remains usable because response removes its local process but not its remote
  leases, siblings, or issued descendants.
expected_system_response: Unaffected separately authorized work continues when the containment decision
  permits it.
expected_adverse: The fenced parent cannot issue a fresh child; all affected gates enforce revocation
```

```
  within the documented liveness bound.
evidence_required:
- approved incident plan
- triage and escalation records
- resource-side fence tests
- evidence custody and recovery authorization records
observed_result: null
```

03.07 — Maintenance

```
family: '03.07'
name: Maintenance
fixture_status: illustrative_not_executed
source_example: EX-MA-01
input: A vendor receives unrestricted shell access to repair a server and retains a reusable administrative
  credential after the service window.
expected_system_response: The approved tool performs the ticketed operation during the authorized window
  and records its outcome.
expected_adverse: An unrelated command or request after expiry is denied even when the technician retains
  a copied ticket.
evidence_required:
- maintenance ticket and approvers
- tool validation record
- technician authorization
- session termination and post-service measurement
observed_result: null
```

03.08 — Media Protection

```
family: '03.08'
name: Media Protection
fixture_status: illustrative_not_executed
source_example: EX-MP-01
input: A retired storage device or exported backup retains sensitive data, even though its online account
  has been revoked.
expected_system_response: Authorized disposition follows the selected sanitization process and verified
  evidence before release.
expected_adverse: A revoked network key alone does not mark the physical media sanitized; missing
↪  verification
  holds release.
evidence_required:
- media inventory and custody
- approved sanitization procedure
- device-specific verification record
- disposition approval
observed_result: null
```

03.09 — Personnel Security

```
family: '03.09'
name: Personnel Security
fixture_status: illustrative_not_executed
source_example: EX-PS-01
input: An employee leaves the organization while previously spawned agents continue receiving renewals
  from a forgotten root.
expected_system_response: An authorized role transfer receives a reviewed narrower admission appropriate
  to the new duties.
expected_adverse: An old worker cannot keep authority alive by replaying a pre-termination eligibility
  assertion.
evidence_required:
- position and screening records
- access agreements
- verified termination event
- account and descendant revocation review
observed_result: null
```

03.10 — Physical Protection

```
family: '03.10'
name: Physical Protection
fixture_status: illustrative_not_executed
source_example: EX-PE-01
input: A well-authenticated root runs in a room with uncontrolled entry, unreliable power, or no tested
  response to fire and cooling failure.
expected_system_response: A permitted operator enters under the facility’s physical process and the
↪  hosting
  evidence remains current.
expected_adverse: A cloned digital badge record does not authorize door entry; a validated host alarm
  prevents new sensitive admissions pending review.
evidence_required:
- access roster and visitor records
- alarm and response exercise
- power/failover maintenance records
- facility control-provider assessment
observed_result: null
```

03.11 — Risk Assessment

```
family: '03.11'
name: Risk Assessment
fixture_status: illustrative_not_executed
source_example: EX-RA-01
input: A team treats a large geometric key representation as a complete answer to business impact,
↪  exploitable
  software defects, privacy harms, and operational exposure.
expected_system_response: Treatment is verified against the affected route and the assessment records
  the residual risk decision.
expected_adverse: A risk is not closed solely because another layer has a strong mathematical proposition.
evidence_required:
- categorization rationale
- threat and privacy-risk assessments
- vulnerability and supplier findings
- approved treatment and retest evidence
observed_result: null
```

03.12 — Security Assessment and Monitoring

```
family: '03.12'
name: Security Assessment and Monitoring
fixture_status: illustrative_not_executed
source_example: EX-CA-01
input: A diagram is treated as proof that every implemented resource enforces the diagram, even though
  a direct database credential bypasses the protected broker.
expected_system_response: A complete reviewed evidence package supports an explicit authorization decision
  for the measured boundary.
expected_adverse: A missing test result or unexplained bypass prevents the implementation from being
↪  reported
  as verified.
evidence_required:
- assessment plan
- examine-interview-test records
- plan of action and milestones
- signed authorization decision and monitoring plan
observed_result: null
```

03.13 — System and Communications Protection

```
family: '03.13'
name: System and Communications Protection
fixture_status: illustrative_not_executed
source_example: EX-SC-01
input: An authenticated session is redirected to an unauthorized endpoint, or applications bypass the
  broker through raw sockets while sharing reusable secrets.
expected_system_response: A fresh correctly bound request reaches the approved peer and verifies the
↪  response
```

```
  direction and sequence.
expected_adverse: The redirect or reused channel record fails before the unauthorized endpoint receives
  sensitive payload.
evidence_required:
- network and crypto profile
- peer/channel binding tests
- broker-bypass and process-isolation tests
- resource and denial-of-service capacity exercises
observed_result: null
```

03.14 — System and Information Integrity

```
family: '03.14'
name: System and Information Integrity
fixture_status: illustrative_not_executed
source_example: EX-SI-01
input: A measured application remains vulnerable after a defect is discovered, or malformed arguments
  exploit a mismatch between policy validation and the actual service operation.
expected_system_response: The fixed adapter passes both ordinary operation and regression cases before
  controlled admission.
expected_adverse: The malformed path is rejected consistently; a signed vulnerable predecessor cannot
  bypass its revocation state.
evidence_required:
- input and error-handling tests
- monitoring and integrity records
- vulnerability triage and root-cause analysis
- patch validation and rollback tests
observed_result: null
```

03.15 — Planning

```
family: '03.15'
name: Planning
fixture_status: illustrative_not_executed
source_example: EX-PL-01
input: The deployment protects one broker but the documented system boundary omits a backup export account
  and an administrative database route.
expected_system_response: Every usable route in the accepted boundary is assigned to an approved
↪  enforcement
  point and owner.
expected_adverse: The unmediated route opens a planning and implementation finding rather than
↪  disappearing
  from the coverage denominator.
evidence_required:
- current system security and privacy plans
- boundary and data-flow review
- user rules acknowledgment
- control selection and tailoring record
observed_result: null
```

03.16 — System and Services Acquisition

```
family: '03.16'
name: System and Services Acquisition
fixture_status: illustrative_not_executed
source_example: EX-SA-01
input: A correctly signed component is purchased or deployed without security requirements, reliable build
  evidence, testing obligations, or a resilient upgrade plan.
expected_system_response: A reviewed release package can progress through staged replacement with
↪  regression
  evidence.
expected_adverse: The signed but incomplete acquisition package is held until its required assurance
↪  deliverables
  are provided and accepted.
evidence_required:
- contract security requirements
- development and design review records
- independent and developer testing
```

```
- approved release and resilience/update evidence
observed_result: null
```

03.17 — Supply Chain Risk Management

```
family: '03.17'
name: Supply Chain Risk Management
fixture_status: illustrative_not_executed
source_example: EX-SR-01
input: An untrusted supplier component enters the signing pipeline, or a formerly approved supplier keeps
  a standing maintenance route after its relationship ends.
expected_system_response: A reviewed supplier artifact with accepted provenance and required tests can
  receive admission.
expected_adverse: A signature alone does not override failed supplier authorization or missing provenance.
evidence_required:
- supplier inventory and risk assessment
- acquisition and notification agreements
- inspection/provenance records
- supplier offboarding and component-disposal evidence
observed_result: null
```

I.4.18. Enhanced CUI protection against persistent threats

SP 800-172 Revision 3, published in May 2026, supplies enhanced requirements selected and required by federal agencies for the described CUI environments. The selected overlay is imported into the same requirement set rather than automatically treating every enhancement as universally required. The official publication is authoritative; dataset differences must be resolved against that publication. [E4]

The HTTP6 reference architecture offers useful design connections for a selected enhanced profile: narrowly scoped administrator ceremonies, independently appraised workload admission, brokered effects, authenticated inventory, bounded delegation, tightly controlled data destinations, detection of abnormal lineage behavior and recovery that creates new generations. Human procedures and defensive operations determine how these mechanisms are used against persistent adversaries. The following is an original workflow example, not a substitute for any enhanced requirement's exact text.

```
example_id: CUI-ENHANCED-01
fixture_status: illustrative_not_executed
selected_overlay: organization-approved-SP800-172r3-selection-pending
asset: critical-program-reporting-enclave
event:
  compromised_worker: reporting-worker-17
  attempted_effect: admit-an-unmeasured-grandchild-on-a-new-host
expected:
  admission: deny
  containment: authorized-lineage-fence
  investigation: preserve-birth-proof-policy-and-outcome-evidence
  recovery: new-approved-generation-after-required-review
required_completion:
  - selected-enhanced-requirement-inventory
  - corresponding-assessment-objectives
  - actual-enforcement-tests
  - threat-hunting-and-incident-procedures-where-selected
observed: null
```

The result is a route to enhanced coverage whose scope remains visible. Claiming all selected enhanced requirements requires evidence for every one of them, including obligations that do not involve authentication. The human pilot light is the accountable beginning of automated authority, not a replacement for every other defensive function.

I.5. Lifecycle, organizational maturity, privacy and agent risk

I.5.1. All seven RMF steps

The RMF lifecycle gives the deployment an accountable route from preparation to continuing operation. The HTTP6 reference architecture supplies artifacts and enforcement at each step; the appropriate officials retain categorization, selection, assessment and authorization responsibilities. The examples below use the seven steps of SP 800-37 Revision 2. [E5]

```
fixture_status: illustrative_not_executed
steps:
- step: Prepare
  input: A reporting enclave is proposed
  expected: Assign mission owners, boundaries, resources, risk strategy and common-control providers
  evidence_required: Preparation and accountability records
  observed_result: null
- step: Categorize
  input: The enclave processes several information types
  expected: Document confidentiality, integrity and availability impact and its approved categorization
  evidence_required: Categorization record and rationale
  observed_result: null
- step: Select
  input: The category and requirements are established
  expected: Select controls, enhancements, overlays and all organization-defined values
  evidence_required: Approved baseline and resolved requirement inventory
  observed_result: null
- step: Implement
  input: Brokers and resources are deployed
  expected: Install the specified enforcement and supporting organizational services
  evidence_required: Implementation statements and exact deployment manifest
  observed_result: null
- step: Assess
  input: The implemented boundary is ready for review
  expected: Evaluate selected objectives using justified examination, interview and test procedures
  evidence_required: Assessment results and remediation records
  observed_result: null
- step: Authorize
  input: The assessment package is submitted
  expected: The designated official makes and records the risk-based authorization decision
  evidence_required: Authorization decision, conditions and accountable official
  observed_result: null
- step: Monitor
  input: A policy adapter or environment changes
  expected: Reassess impacted requirements, refresh evidence and enforce ongoing decisions
  evidence_required: Monitoring records and change-triggered assessment results
  observed_result: null
```

An authorization decision is not the same object as a cryptographic root-admission proof. The former is an official risk decision for a system and its conditions. The latter activates an instance under a specific runtime policy. A deployment can require an approved authorization decision before root admission; the two records remain separately identifiable and appropriately authorized.

I.5.2. All four SSDF groups

The selected SSDF baseline is final Version 1.1, SP 800-218. The four groups below organize development evidence around the artifacts that eventually become admitted code and policy. Draft revisions are tracked as proposed changes rather than silently substituted into the selected baseline. [N15; E10]

```
fixture_status: illustrative_not_executed
groups:
- group: PO — Prepare the Organization
  input: A new policy engine team starts work
  expected: Establish roles, requirements, secure environments and review criteria
  evidence_required: Training, secure development plan and environment records
  observed_result: null
```

```
- group: PS — Protect the Software
  input: A release artifact is copied between build and production
  expected: Protect source, build inputs and release integrity and retain provenance
  evidence_required: Repository access, release integrity and protected archive evidence
  observed_result: null
- group: PW — Produce Well-Secured Software
  input: The canonical request parser accepts ambiguous encodings
  expected: Design, review and test parsing and policy behavior before releasing the adapter
  evidence_required: Threat model, code review, parser tests and defect disposition
  observed_result: null
- group: RV — Respond to Vulnerabilities
  input: A broker bypass is discovered after deployment
  expected: Assess impact, remediate, perform root-cause analysis and issue a controlled replacement
  evidence_required: Vulnerability triage, fix, regression test and response records
  observed_result: null
```

For a Policy Decision Point, these practices include threat modeling the compiler and normalizer, protecting the build system, reviewing trusted key-use APIs, testing ancestry intersection and correcting defects in deployed adapters. Immutable identities make releases traceable; secure development establishes whether a release deserves admission. A signed defect remains a defect until the implementation is corrected and its affected requirements are reassessed.

I.5.3. CSF Tiers describe organizational practice, not circuit strength

CSF 2.0 Tiers distinguish characteristics of organizational cybersecurity risk governance and management. They are not cryptographic key sizes, identity assurance levels, or product certificates. The examples are illustrative organizational states used in profile planning. [N8]

```
example_group: CSF-TIERS
fixture_status: illustrative_not_executed
cases:
  - tier: "1 — Partial"
    input: "A team installs one lineage broker without an agreed enterprise risk process"
    expected_action: "Establish ownership and a current profile; record actual practices"
    evidence: "Observed decisions and governance records"
  - tier: "2 — Risk Informed"
    input: "Management approves risks, but practices vary across teams"
    expected_action: "Use approved risk priorities to standardize admitted profiles"
    evidence: "Management decisions and deployment consistency review"
  - tier: "3 — Repeatable"
    input: "Roles, assessment and change procedures are consistently maintained"
    expected_action: "Operate the repeatable lifecycle and retain evidence across deployments"
    evidence: "Institutional policy and repeated execution records"
  - tier: "4 — Adaptive"
    input: "New threat and incident evidence changes the risk response"
    expected_action: "Update practices and approved profiles through accountable learning"
    evidence: "Measured feedback, approved adaptations and resulting outcomes"
observed_results: null
```

I.5.4. Privacy Framework: five explicitly selected Version 1.0 functions

The following extension deliberately names Privacy Framework Version 1.0 as its selected reference. Its five functions address privacy risks arising from data processing, beyond whether an attacker can recover an authentication secret. Lineage logs, enrollment evidence, identifiers and inference about people's activities therefore receive their own privacy analysis and operational controls. [E11]

```
fixture_status: illustrative_not_executed
functions:
- function: Identify-P
  input: A lineage log links people to sensitive workflows
  expected: Inventory the data processing and evaluate privacy risks to individuals
  evidence_required: Data map, purpose and privacy risk assessment
  observed_result: null
```

```
- function: Govern-P
  input: Managers propose a new secondary use for audit histories
  expected: Obtain appropriate privacy governance review and define permitted purposes
  evidence_required: Approved purpose, responsible roles and review decision
  observed_result: null
- function: Control-P
  input: A person makes an applicable correction or deletion request
  expected: Route it through an authorized process with identity checks, retention rules and downstream
    handling
  evidence_required: Request handling, applicable exceptions and outcome evidence
  observed_result: null
- function: Communicate-P
  input: A service starts retaining new device identifiers
  expected: Deliver clear notice and appropriate interaction or redress mechanisms
  evidence_required: Notice content, delivery and complaint-handling records
  observed_result: null
- function: Protect-P
  input: An analyst requests raw identifiers beyond an approved investigation
  expected: Apply scoped access, minimization, protected storage and authorized disclosure
  evidence_required: Privacy-aware access and disclosure tests
  observed_result: null
```

The source's conditional secrecy theorem remains valuable for the specified observation model. It does not establish that all lawful processing is appropriate, that audit identities are unlinkable, or that every data-subject obligation has been performed. The integration instead provides purpose-bound authorization, minimized logging, restricted analyst roles and accountable processing procedures. Privacy officers select the applicable requirements and assess the actual effects on individuals.

I.5.5. AI RMF: four functions for delegated agents

The selected AI RMF 1.0 Core comprises GOVERN, MAP, MEASURE and MANAGE. The following examples connect those functions to agent execution while preserving the distinction between permission and correctness. An authorized agent can still produce an inaccurate output; a narrow capability does not establish validity, fairness or suitability for an intended decision. Those outcomes require their corresponding evaluations and oversight. [E12]

```
fixture_status: illustrative_not_executed
functions:
- function: GOVERN
  input: An autonomous agent is proposed for financial reporting
  expected: Assign accountable owners, approved uses, oversight and escalation policy
  evidence_required: AI policy, ownership and human-oversight records
  observed_result: null
- function: MAP
  input: The agent can both inspect and publish a report
  expected: Map affected parties, data, tools, intended uses and downstream harms
  evidence_required: Use-context and impact assessment
  observed_result: null
- function: MEASURE
  input: The agent stays in scope but writes an incorrect financial conclusion
  expected: Evaluate validity, reliability, relevant fairness, privacy and security separately from access
    permission
  evidence_required: Task-specific evaluation, human review and performance evidence
  observed_result: null
- function: MANAGE
  input: Testing identifies an unacceptable tool or decision risk
  expected: Restrict delegated capabilities, require review or suspend the deployment and reassess
  evidence_required: Risk treatment, restricted policy and retest evidence
  observed_result: null
```

The strongest contribution of the HTTP6 reference architecture is that a model's proposal remains a request until an independent monitor authorizes the specific effect. A prompt injection cannot rewrite a committed policy merely by persuading the model. The broader AI risk program evaluates what happens inside the permitted task: data suitability, reliability, meaningful human review and

appropriate intervention. This preserves both security containment and responsible evaluation of authorized behavior.

Appendix J. Assessment dossier, source inventory, and evidence exchange

J.1. Review-specific submission manifest

This is an architecture and implementation-review dossier. The receiving program and selected system boundary determine the actual mandatory submission materials. A cryptographic-module validation, an agency system authorization, a standards contribution, and a research-paper review are different processes. This dossier provides the common technical structures and records the additional artifacts needed for the chosen route. It does not identify a single universal checklist called “NIST review.” [N2; N13; N14; E5]

Artifact	Contents and responsible role	Material in this edition
Architecture specification	Components, actors, interfaces, boundaries, owner	Chapters 1–5; Appendices A and D
Private-reference construction dossier	Enrollment, witness relation, public view, composition and exposure analysis	Conditional proof and interface in Appendix B; production algorithm assigned to construction owner
System security and privacy plan	Actual boundary, data flows, selected controls, providers, implementation statements	Structured draft below; system owner completes deployment facts
Assessment plan	Source objectives, methods, scope, subjects, expected evidence	Draft plan and raw cases in G–I
Assessment results	Observations, findings, risks, exact environment, reviewer	Local reference-model results in K; production results remain separate
Corrective-action register	Finding, owner, milestones, verification, status	Draft record below
Cryptographic profile	Module scope, versions, certificates, approved services, entropy and key lifecycle	Checklist in I and below; actual module evidence assigned
Source and build package	Actual engine and adapters, reproducible inputs, dependency lock, measurements	Reference code and assembly tools; original production engine assigned
Test vectors and models	Canonical encoding, signed records, positive/adverse cases, finite state bounds	E, F, K and models/
Operational plans	Enrollment, incident, recovery, training, facilities, supplier and privacy procedures	Family cases and owner assignments; actual organizational records assigned
Provenance manifest	Original source hashes, generated artifact hashes, version trail	Appendix L and `manifest.json`

J.2. Draft system implementation record

```
record_type: http6-review-ssp-draft
exchange_status: custom-record-not-an-OSCAL-SSP
system:
  id: reporting-enclave-example
  owner: system-owner-to-be-appointed
  boundary_status: illustrative-pending-approval
  protected_resources: [ledger-view, aggregate-report-store, mint-service]
  authoritative_state: [birth-registry, nonce-ledger, policy-registry]
  trust_anchors: [human-admission-verifier, runtime-appraiser, evidence-provider-roots]
  prohibited_paths: [direct-db-credential, raw-egress-socket, arbitrary-signing-api]
  selected_baseline: null
  impact_categorization: null
  requirement_profile_digest: null
components:
  - name: resource-gate
    proposed_controls: [AC-3, AC-25, IA-9, SC-23]
    implementation_ref: production-adapter-pending
    responsibility: system-specific
  - name: facility-provider
    proposed_controls: [PE-2, PE-3, PE-11, PE-13]
    implementation_ref: applicable-provider-evidence-pending
    responsibility: inherited-and-customer
parameters_resolved: false
approval: null
```

The record becomes a system plan after the owner supplies the real boundary, selected obligations, implemented components, inherited responsibility agreements, and actual parameter values. The identifiers above are illustrative selections rather than an approved baseline. Discovery of an alternate path creates a change to the boundary and its enforcement assignments, not a silent exclusion.

J.3. Draft assessment plan and result linkage

```
record_type: http6-review-assessment-plan
exchange_status: custom-record-not-an-OSCAL-assessment-plan
scope_ref: approved-system-plan-required
assessment_id: reporting-gate-review
source_objectives: actual-selected-53A-objective-identifiers-required
methods: [examine, interview, test]
subjects:
  - admission-verifier
  - minting-broker
  - resource-gate
  - durable-state-service
  - responsible-operators
cases:
  - id: replay-race
    positive: one-valid-proof-produces-one-controlled-outcome
    adverse: duplicate-proof-produces-no-second-effect
    inspect: [ledger, resource-outcome, authenticated-retry]
  - id: ancestry-expansion
    positive: narrow-child-reads-approved-view
    adverse: child-cannot-export-outside-root-ceiling
    inspect: [signed-chain, policy-decision, actual-data-destination]
results:
  observation_refs: []
  finding_refs: []
  risk_refs: []
  implementation_version: null
  assessor: null
  determination: not_assessed
```

The result record retains expected and observed behavior separately. A completed test specifies who collected the evidence, when, under which software/policy/hardware configuration, and which actual effect was inspected. Finding disposition and risk acceptance remain distinct from verification that a safeguard is operating. [N2]

J.4. Corrective action and integration assignments

```
record_type: http6-review-corrective-action
exchange_status: custom-record-not-an-OSCAL-POAM
item_id: INTEGRATION-RW-01
origin: source-implementation-deliverable
requirement_or_claim: binary-conditioned-sequential-privacy
owner: construction-implementation-owner
required_artifacts:
  - exact-enrollment-and-witness-source
  - binary-field-and-private-map-inventory
  - complete-observer-view
  - construction-specific-composition-proof
  - fresh-randomness-or-computational-profile
  - implementation-test-vectors
status: pending-production-artifacts
milestones: []
observed_result: null
reviewer: null
```

This item is a review deliverable, not a reported security defect. Equivalent assignments identify real hardware appraisal, key custody, external effects and ledger consistency, resource coverage, identity lifecycle, organizational procedures, and independent assessment. The record identifies what evidence would support closure instead of hiding incomplete source access behind a generic “passed” label.

J.5. OSCAL exchange map

NIST OSCAL separates the catalog and profile, component and system implementation, assessment plan, assessment results, and plan of action and milestones. The following mapping describes how the custom architecture records can populate the appropriate official model. It does not claim that the custom YAML snippets already validate against those schemas. [R2]

Architecture review object	Intended OSCAL relationship	Export acceptance
Authoritative selected requirement sources	Catalog and profile	Preserve source control identifiers and resolve imports
Gate, mint, key service, common provider	Component definition	Bind exact component and applicable implementation statements
System boundary and responsibilities	System security plan	Use selected profile and actual control implementation
Tests, methods, subjects, objectives	Assessment plan	Reference the assessed system plan and exact objectives
Observations, findings, reviewed outcomes	Assessment results	Link to the plan, collected evidence, and determinations
Unresolved assignments and findings	Plan of action and milestones	Identify actual items, responsible owners, milestones and disposition

An exporter pins the OSCAL version, validates against its official schema, resolves imported references, and reviews the meaning of each populated field. Schema-valid data is structured data, not independent evidence that its statements are true. Production exports are an assigned integration artifact; this edition supplies the information model, example records, and source-import tool.

J.6. Independently obtained source inventory

The full official SP 800-53 OSCAL catalog was not imported into this package. The source repository and publication pages were consulted, but the large raw catalog could not be fetched into the build environment. Accordingly, the deliverable includes a local-file importer and synthetic importer tests rather than a fabricated complete catalog. The original twenty-family and CSF outcome case sets are fully preserved in G and H.

Run the importer only against an independently obtained official catalog with a reviewed digest and exact metadata version:

```
python reference_code/import_oscal_catalog.py catalog.json \
  --sha256 ACTUAL_REVIEWED_SOURCE_SHA256 \
  --version ACTUAL_CATALOG_METADATA_VERSION \
  --output source_inventory.json
```

The importer preserves original controls, enhancements, statement parts, parameters, properties, and group paths. It leaves selection approval, assessment-procedure sources, implementation assignments, and observations unresolved. A complete source catalog is then tailored under the actual boundary and approvals; the corresponding assessment objectives are imported separately where needed. The resulting approved inventory, rather than a manually counted family list, supplies the denominator for requirement closure. [E8; E9; N16]

J.7. Cryptographic and private-reference evidence checklist

The deployment record names the entropy sources and RBG construction, key-generation services, public-key and symmetric algorithms, parameter sets, cryptoperiods, generation/rotation/erasure procedures, and export restrictions. For every claimed validated service it supplies the actual certificate, exact module version, operational environment, security policy, approved mode, and service invocation. The Rosario–Wang protocol receives its own characterization; an underlying module certificate covers its stated module boundary and services. [N10–N14; R3]

The privacy profile additionally identifies whether its result is exact or computational, where private interpretation resides, what a copied binary reveals, how fresh masks or keys are provisioned, and what happens after state exposure. The review tests successful-operation secrecy, active-query soundness, recovery behavior, and side-channel scope separately. The approved trusted-display/input path is part of the implementation, not merely a feature assumed from the appearance of a screen.

J.8. Program-specific review gate

Before actual submission, the responsible owner selects the receiving program, intended assessment boundary, applicable editions, confidentiality handling, review format, and required evidence depth. The architecture paper and reference source are suitable technical materials for that decision. A system authorization adds its implemented plan and actual assessment; a module validation adds its laboratory and module documentation; a construction review adds its full protocol and security argument. Each route has an explicit owner, artifact list, and approval record.

Appendix K. Executed reference checks and production acceptance plan

K.1. Actual bounded observations in this edition

The new authority and importer source tests completed **47 pytest cases**. The retained completeness checker completed **18 synthetic record cases**. The finite state explorer visited **240 states and 720 transitions** within its declared domain. The finite privacy program performed exact distribution comparisons for one through six ceremonies, plus disclosed-mask, deterministic-partition, and reused-mask diagnostics. The public signed-record replay produced one local effect and rejected its second submission. These results concern the included reference artifacts, not a production HTTP6 reference architecture deployment or execution of a trillion logins.

Reference source tests

```
.............................................                            [100%]
47 passed in 0.48s
```

Existing record checker

```
18 synthetic checker tests passed; no HTTP6 implementation assessment performed.
```

Exact finite privacy illustrations

```
{
  "fresh_private_mask_rounds_1": true,
  "fresh_private_mask_rounds_2": true,
  "fresh_private_mask_rounds_3": true,
  "fresh_private_mask_rounds_4": true,
  "fresh_private_mask_rounds_5": true,
  "fresh_private_mask_rounds_6": true,
  "disclosed_mask_view_distinguishes_secrets": true,
  "public_deterministic_partition_path_identifies_secret": true,
  "reused_mask_cancels_in_xor": true
}
```

Finite state exploration

```
{
  "model": "finite-attenuation-and-consumption-v1",
  "states": 240,
  "transitions": 720,
  "invariants": "all checked in this finite model",
  "bounds": {
    "max_instances": 3,
    "permission_bits": 2,
    "challenge_positions": 2
  },
  "production_assessment": false
}
```

Public signed example

```
{
  "fixture_kind": "synthetic_only",
  "first": "permitted_once_in_local_model",
  "replay": "rejected",
  "local_effect_count": 1
}
```

Schema checks

```
{
  "original_policy_schema": "valid",
  "original_policy_example": "valid",
  "review_schema": "valid",
  "review_examples_validated": [
    "review_policy",
    "review_request",
    "review_root_grant",
    "review_child_grant",
    "review_action_proof",
    "review_evidence_record"
  ],
  "meaning": "Syntax checks only; no production security certification."
}
```

Recorded environment

```
{
  "python": "3.13.5",
  "packages": {
    "cryptography": "46.0.4",
    "jsonschema": "4.26.0",
    "PyYAML": "6.0.3",
    "pytest": "9.0.2"
  },
  "cryptographic_validation_claim": false,
  "environment_kind": "local-reference-review"
}
```

K.2. Reproduce the local source checks

```
python -m venv .venv
. .venv/bin/activate
python -m pip install -r requirements.txt
cd reference_code
python -m pytest -q test_authority_model.py test_catalog_import.py
python test_closure_checker.py
python privacy_model.py
python state_model.py
python run_public_example.py
```

The lock file records the actual package versions in the build environment. The tests use temporary local databases and synthetic keys. They perform no network operations, contact no production service, and access no user's private reference.

K.3. Production acceptance campaign

Each case below names the enforcing boundary and the independent evidence needed for deployment assessment. Observations remain empty until the actual system is exercised. The full original cross-control campaign follows.

```
campaign_id: CROSS-CONTROL-ACCEPTANCE
fixture_status: illustrative_not_executed
cases:
  - id: TEST-REPLAY-RACE
    setup: "One valid action, one epoch and nonce, concurrent requests through two replicas"
    authorized_path: "One consumption and one durable operation outcome"
    adverse_path: "Duplicate submissions create no second effect"
    inspect: [ledger-state, business-effect, retry-response]
  - id: TEST-POLICY-MUTATION
    setup: "Admitted worker with pinned role and policy catalog"
    authorized_path: "Approved aggregate report succeeds"
    adverse_path: "Edited local catalog does not authorize raw export"
    inspect: [admission-digest, inherited-decision, actual-destination]
```

```
  - id: TEST-ANCESTOR-REMOVAL
    setup: "Grandchild whose root forbids ledger deletion"
    authorized_path: "Permitted read verifies complete ancestry"
    adverse_path: "Removing the root restriction breaks the chain or fails the inherited predicate"
    inspect: [delegation-records, verifier-decision, database-effects]
  - id: TEST-INSTANCE-CLONE
    setup: "Admitted instance and a copied image on another generation"
    authorized_path: "Current registered instance proves its identity"
    adverse_path: "The copy obtains no authority from disk bytes alone"
    inspect: [attestation, protected-key-binding, generation-fence]
  - id: TEST-DESTINATION-SWAP
    setup: "Allowed hostname with a redirect or changed resolved peer"
    authorized_path: "The admitted peer and approved operation succeed"
    adverse_path: "Unapproved destination or operation is rejected after normalization"
    inspect: [resolved-address, authenticated-peer, canonical-request]
  - id: TEST-BUDGET-RACE
    setup: "Sibling workers concurrently reserve the last remaining allocation"
    authorized_path: "Only admissible remaining allocations commit"
    adverse_path: "Aggregate consumption never exceeds the parent budget"
    inspect: [reservations, consumption, restart-history]
  - id: TEST-HUMAN-TIMEOUT
    setup: "Human session expires while machine heartbeats continue"
    authorized_path: "A fresh required human ceremony renews the human session"
    adverse_path: "Machine activity does not manufacture human reauthentication"
    inspect: [human-auth-time, workload-lease, session-decision]
  - id: TEST-PARENT-FENCE
    setup: "Authorized operator retires a scheduler epoch"
    authorized_path: "Unaffected approved lineages continue if policy permits"
    adverse_path: "Affected descendants stop within the admitted propagation and lease bound"
    inspect: [issuer-renewals, resource-gates, last-accepted-action]
  - id: TEST-RECOVERY-ROLLBACK
    setup: "Restore an image containing consumed positions"
    authorized_path: "New admitted generation resumes the permitted business task"
    adverse_path: "Old positions remain consumed and old keys do not regain authority"
    inspect: [durable-ledger, birth-record, transaction-outcome]
  - id: TEST-PROVIDER-EVIDENCE
    setup: "A selected hosting or training dependency has expired evidence"
    authorized_path: "A valid scoped provider record supports the assigned decision"
    adverse_path: "Expired or wrong-scope evidence cannot close the requirement"
    inspect: [issuer, scope, validity, owner-review]
observed_results: null
```

K.4. Additional construction and integration cases

```
additional_production_tests:
  - id: ARTIFACT-POSSESSION
    action: enumerate every field recoverable from the actual binary
    inspect: secret exposure, private maps, seeds, verifier tables, snapshots
    observed: null
  - id: FULL-VIEW-PRIVACY
    action: compare the formally specified complete sequential distributions
    inspect: witnesses, displays, results, timing model, prior history, randomness
    observed: null
  - id: ACTIVE-QUERY-SOUNDNESS
    action: exercise rate-limited chosen attempts and invalid-context responses
    inspect: guessing bounds, oracle behavior, error handling, replay and reset policy
    observed: null
  - id: HARDWARE-KEY-BINDING
    action: attempt use from a copied image and a new machine generation
    inspect: genuine attestation, non-exportable key use, current registry and fencing
    observed: null
  - id: MINT-TRANSACTION-RACE
    action: race child requests against revocation and the final birth allocation
    inspect: atomic birth record, reservations, signed lineage, no duplicate authority
    observed: null
  - id: EXTERNAL-EFFECT-RECONCILIATION
    action: interrupt before and after downstream acceptance
    inspect: outbox, receiver idempotency, reconciliation, actual business outcome
    observed: null
  - id: SIDE-CHANNEL-PROFILE
```

```
    action: evaluate the selected runtime and interaction leakage channels
    inspect: timing, memory exposure, power where applicable, debug and diagnostic paths
    observed: null
```

A construction proof and these deployment observations address different questions. The reviewer binds both to the same selected version and threat profile. Evidence discovered to be stale, mismatched, or incomplete opens the corresponding review item rather than being reused under a different implementation name.

Appendix L. Provenance, changes, and reproducible assembly

L.1. Source basis and preservation

The uploaded public paper and expanded mapping are preserved byte-for-byte under `sources/`. Earlier architecture manuscripts, the complete original policy schema/example, and the preceding example catalogs and checker are also retained. The expansion replaces the short public introduction with exactly 2,000 prose words, retains the original main equations, adds binary-conditioned privacy and minting detail, adds a concrete review pattern to each remaining chapter, and embeds the detailed supplementary material as appendices.

The new definitions and propositions are labeled [D1]; new reference source, schemas, and assembly specifications are labeled [D2]. Their provenance is this review edition rather than a claim that they were found in an existing production repository. The three implementation profiles remain distinct: the source architecture, the narrow executable review model, and the proposed production integration contract.

`ENI6MA_NIST_Public_White_Paper(1).md`
SHA-256: d28472def34e4fbb54d0bcbf43c38de6e75be46ce8b243a1101854626bd36299.

`ENI6MA_NIST_Requirements_and_Control_Mapping_Enhanced(1).md`
SHA-256: 4e43da6471ae87bd888f62d46da9a6ae104f392f6179925b3c8403bb8fb173ad.

ENI6MA_Unified_Technical_Manuscript.md
SHA-256: 22347869d7200c5aa89b9d29927548b643337372ac33ca0e26c55a074090b24c.

ENI6MA_White_Paper_Human_Originated_Identity.md
SHA-256: af112a7f5678060ab2498288e9438f0d9ff50cc22391679946ac76366f960d98.

L.2. Review changes that preserve technical meaning

The binary-possession claim is now conditioned on the actual acquired bytes and complete subsequent observations. The paper distinguishes zero incremental leakage from independent secrecy of the acquired artifact itself. It states the trillion-observation conclusion as a finite-sequence theorem under its explicit premises and retains the source's separate exact and computational profiles.

The hereditary model preserves distinct instance identity, authenticated parent links, explicit mint rights, template/depth/lifetime limits, policy conjunction, current state, and independent resource enforcement. It does not replace those mechanisms with an inference from transcript secrecy. The review fixtures retain empty production observation fields, while executed local model checks carry their actual results and bounds.

The new reference encoding is named `TEST-CANON-1` rather than represented as full JCS. The SQLite example models effects in the same local transaction and does not imply distributed or external exactly-once semantics. The registry helper is explicitly a test fixture rather than genuine attestation. These scope bindings make the code examples reproducible and correctly interpretable.

L.3. Reference verification and source catalog status

Official NIST publication and program pages were consulted for the selected SP 800-53/53A release, SP 800-63 suite, zero-trust guidance, RBG construction, CUI guidance, incident response, sanitization, SSDF publication status, OSCAL model structure, and CMVP validation scope. Selected editions remain named rather than silently replaced. The full bulk official control catalog was not material-

ized; the local-file importer and source selection procedure in J are the delivered import path. The source-derived classification examples are not represented as that full catalog.

L.4. Build and reproducibility

book_manifest.json specifies the chapter and appendix order. build/assemble.py combines those Markdown files, preserving `$...$` and `$$...$$` mathematics. Schemas, review code, examples, models, and evidence remain separately editable. `manifest.json` records file integrity hashes. Rebuilding the PDF requires Pandoc and XeLaTeX with the named installed fonts; font files are not distributed in the package.

The rendered PDF is a typeset representation of the master Markdown. Page, structure, and source checks concern the artifact itself. The package does not use PDF formatting success or reference-model tests as a substitute for the substantive production assessment described in J and K.

References

The following register retains the publication identifiers and selected editions of the supporting mapping. The source mapping is the basis for the classifications and relationships presented here. References to a selected edition establish the paper's scope; they do not silently substitute a different deployment baseline. Source sections and equations provide traceability for the architectural propositions and illustrative cases.

[S1] Rosario, Frank Dylan. *Cryptographic Geometry, Human-Originated Identity, and Hereditary Authority.* Consolidated edition, November 9, 2026. Supporting architecture manuscript. Part I: foundations. Part II: formal theory. Part III: hereditary authority and immutable policy. This source is an architecture manuscript, not an assessment report.

[S2] Rosario, Frank Dylan. *HTTP6 and NIST: Human-Originated Identity, Hereditary Authority, and Security Control Enforcement.* Expanded Edition 2.0, November 9, 2026. Architecture crosswalk, illustrative control examples, and evidence-completion specification. Supplied as Supplement S1 in the submission package.

[N1] NIST. *SP 800-53 Revision 5: Security and Privacy Controls for Information Systems and Organizations*, including the Release 5.2.0 update announced August 27, 2025. Control text consulted in the Rev. 5 publication; release scope checked against the official update. Publication and update links; Release 5.2.0 announcement; Control text. Relevant controls are individually identified in the crosswalk and corresponding sections.

[N2] NIST. *SP 800-53A Revision 5: Assessing Security and Privacy Controls in Information Systems and Organizations*, January 2022, with Release 5.2.0 assessment updates. Chapter 2, particularly assessment methods and objectives; applicable control assessment procedures. Official publication.

[N3] NIST. *SP 800-63B-4: Digital Identity Guidelines — Authentication and Authenticator Management*, July 2025. §§2.2–2.3; 3.2.2; 3.2.5; 3.2.7–3.2.8; 3.2.10; 3.2.12–3.2.13; 4; 5; 7. Complete publication; Authenticator requirements; Session management; Authenticator events.

[N4] NIST. *SP 800-63A-4: Digital Identity Guidelines — Identity Proofing and Enrollment*, July 2025. Introduction, expected proofing outcomes, enrollment, and selected assurance-level requirements. Official text.

[N5] NIST. *SP 800-63C-4: Digital Identity Guidelines — Federation and Assertions*, July 2025. Federation parties, trust agreements, assertions, and assurance requirements. Official text.

[N6] Rose, S., et al. *SP 800-207: Zero Trust Architecture*, August 2020. §§2.1 and 3: zero-trust tenets and logical components. Official publication.

[N7] Chandramouli, R., and Z. Butcher. *SP 800-207A: A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments*, September 2023. Identity-tier and network-tier policy enforcement for distributed applications. The title here follows the final publication's title page. Final publication.

[N8] NIST. *The NIST Cybersecurity Framework (CSF) 2.0*, CSWP 29, February 26, 2024. Appendix A: Core outcomes. Official publication.

[N9] Hu, V., et al. *SP 800-162: Guide to Attribute Based Access Control (ABAC) Definition and Considerations*, January 2014, updated August 2019. Official publication.

[N10] NIST. *SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation*, January 2018. Entropy-source design and assessment. Official publication.

[N11] NIST. *SP 800-90C: Recommendation for Random Bit Generator (RBG) Constructions*, September 2025. Official publication.

[N12] Barker, E. *SP 800-57 Part 1 Revision 5: Recommendation for Key Management — Part 1: General*, May 2020. Key protection, lifecycle, and management. Official publication.

[N13] NIST. *FIPS 140-3: Security Requirements for Cryptographic Modules*, March 2019. Official publication.

[N14] NIST. *Cryptographic Module Validation Program.* Validation scope and module evidence. Program reference.

[N15] Souppaya, M., K. Scarfone, and D. Dodson. *SP 800-218: Secure Software Development Framework (SSDF) Version 1.1*, February 2022. Secure development practices and evidence. Official publication.

[N16] NIST. *SP 800-53B: Control Baselines for Information Systems and Organizations*, October 2020, with Release 5.2.0 issued for consistency and no baseline changes in that release. Baseline selection and tailoring context. Official publication.

[E1] NIST. *FIPS 199: Standards for Security Categorization of Federal Information and Information Systems.* Official publication. Used for confidentiality, integrity and availability impact classification.

[E2] NIST. *FIPS 200: Minimum Security Requirements for Federal Information and Information Systems.* Official publication. Used for the seventeen minimum security areas, within the standard's applicability.

[E3] NIST. *SP 800-171 Revision 3: Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations*, May 2024; and *SP 800-171A Revision 3* assessment procedures. Authoritative HTML publication; Assessment publication. Used for the seventeen CUI requirement-family namespaces and assessment integration.

[E4] NIST. *SP 800-172 Revision 3: Enhanced Security Requirements for Protecting Controlled Unclassified Information*, May 2026. Official publication and selection statement. The enhanced selection is agency- and agreement-specific; the publication identifies the PDF as authoritative over dataset discrepancies.

[E5] NIST. *SP 800-37 Revision 2: Risk Management Framework for Information Systems and Organizations*, December 2018. Official publication; RMF steps. Used for Prepare, Categorize, Select, Implement, Assess, Authorize and Monitor.

[E6] NIST. *SP 800-61 Revision 3: Incident Response Recommendations and Considerations for Cybersecurity Risk Management*, April 2025. Official publication. Used for operational incident-response integration.

[E7] NIST. *SP 800-88 Revision 2: Guidelines for Media Sanitization*, September 2025. Official publication. Used for media lifecycle and sanitization evidence integration.

[E8] NIST. *OSCAL Content Repository.* Official maintained repository. Used as the designated machine-readable import source family, not as a claim that a complete selected catalog was imported into this document.

[E9] NIST. *Cybersecurity and Privacy Reference Tool (CPRT).* Catalog portal. Used as an additional official requirement-data discovery source.

[E10] NIST. *Secure Software Development Framework: Publications.* Official status register. Distinguishes final SP 800-218 Version 1.1 from the listed draft Version 1.2.

[E11] NIST. *Privacy Framework Version 1.0*, January 2020. Versioned publication page; Function and use guidance. Version 1.0 is explicitly selected for this function-level extension.

[E12] NIST. *AI Risk Management Framework 1.0*, January 2023. Official Core text. Used for GOVERN, MAP, MEASURE and MANAGE; outcomes concerning agent correctness and broader trustworthiness remain separately evaluated.

References added for the expanded review edition

[D1] This edition, Appendix B. *Binary-possession privacy and hereditary authority propositions.* Explicit model extensions and deductions from the source's conditional privacy and delegation definitions. Not an independent production security evaluation.

[D2] This edition, Appendices A–F and J–L. *HTTP6 review specifications, schemas, reference source, and bounded checks.* Original review artifacts prepared for this edition; production integration interfaces are separately identified.

[R1] IETF. RFC 8785. *JSON Canonicalization Scheme (JCS).* The production encoding profile may select this standard; the executable review model uses its own restricted TEST-CANON-1 encoding. Official specification.

[R2] NIST. *OSCAL Layers and Models; Assessment Layer; System Security Plan Model.* Official descriptions of catalog, profile, component, system-plan, assessment-plan, assessment-results, and POA&M model relationships. Layers and models; assessment layer. Consulted November 9, 2026.

[R3] NIST. *Cryptographic Module Validation Program: Validated Modules.* Validation is tied to the actual module, its configuration, and documented services; implementing an approved function alone is not module validation. Official program guidance. Consulted November 9, 2026.

[R4] Python Cryptographic Authority. *Cryptography: Elliptic Curve Cryptography.* API documentation for key generation, ECDSA signing, and signature verification used by the local reference model. Official documentation. The tested installed library version is recorded in Appendix K.