Clinical Truth Chain (CTC)

AAS Constitution (v1.0)

ADMISSIBLE AUTOMATION STANDARD (AAS) — CORE NORMATIVE SPECIFICATION

Author: Yong Bok Lee
Organization: Meridian Verity Group LLC
Contact: MeridianVerity@proton.me

Release date: 1.23.26
DOI: 10.5281/zenodo.18340955

Not legal advice / not medical advice: This document is informational and technical; it does not constitute legal, medical, coding, compliance, reimbursement, or regulatory advice.
Non-limiting notice: Examples, fields, schemas, thresholds, mappings, and workflows are illustrative; implementations may vary.
Claims-control notice: Nothing in this document limits claim scope; patent claims control.

Vendor-neutral does not mean IP-unencumbered; implementers must obtain any necessary rights.
Normative language: RFC 2119 / RFC 8174 keywords (MUST, SHOULD, MAY, etc.) are interpreted only when appearing in all capitals.

Preamble

We hold a single operational principle to be non-negotiable in systems that touch human lives and public money: decisions that deny, delay, pay, or claw back care MUST be provable. When a decision cannot be replayed from machine-verifiable evidence under the declared policy, it is not admissible automation.

This Constitution defines Admissible Automation as a safety property: fail-closed gates that require replay-verifiable receipts and continuity-checked anchoring before regulated or financially consequential actions proceed.

Its purpose is not to prescribe one vendor, architecture, or policy—only to make outcomes checkable by independent verifiers with deterministic reason codes and correction semantics that do not rewrite history.

Contents

- 1. Purpose
- 2. Scope and actors
- 3. Normative keywords
- 4. Definitions
- 5. Safety property: Fail-Closed Admissibility
- 6. Architectural model: five rails and evidence spine
- 7. Normative invariants
- 8. Receipt requirements (canonicalization, signatures, policy pinning)
- 9. Continuity-Verified Log (CVL) requirements
- 10. Replay verification algorithm (normative)
- 11. Deterministic failure semantics (reason codes)
- 12. Correction without rewriting history (AOR / CR)
- 13. Conformance artifacts (validator, badge profiles, test vectors)
- 14. FHIR interoperability constitution (DocumentReference + Provenance + AuditEvent)
- 15. Security considerations
- 16. Privacy considerations
- 17. Governance, change control, and registries
- Appendix A. Minimal receipt schema checklist (informative)
- Appendix B. Minimal FHIR bundle patterns (informative)
- Appendix C. Example Conformance Result (informative)
- Appendix D. Reason code registry (normative baseline)
- Appendix E. Canonicalization profile guidance (informative)

1. Purpose

This Constitution defines the Admissible Automation Standard (AAS): a normative, implementable rule for when automation in clinical and revenue-cycle workflows is allowed to take effect.

AAS converts narrative compliance (“we followed policy”) into machine-verifiable evidence (“here are the receipts, proofs, and replay verification outputs”).

The intended outcome is procurement- and audit-grade determinism: independent verifiers can reproduce an adjudication-relevant decision (or reproduce why it must fail-closed) using canonical receipts, pinned policies, and continuity-verified anchoring.

2. Scope and actors

2.1 In scope (non-limiting)

- Prior authorization (PA) decisioning, including guardrails, pend/deny outcomes, and appeal/overturn workflows.
- Authorization-to-claim binding and adjudication of claim line items.
- AI-influenced clinical Decision Support Intervention (DSI) governance and permit-before-action semantics.
- AI-influenced billing events and permit-before-bill gating (including configured anti-circumvention).
- Correction workflows (appeals, recoupments, refunds, reclassifications) represented as forward-linked receipts without rewriting history.
- Independent replay verification by payers, auditors, regulators, or other authorized relying parties.

2.2 Out of scope / non-goals (reviewer-relaxers; non-limiting)

- This Constitution does NOT define medical necessity rules, coverage policy content, reimbursement rates, or coding guidance; it defines how such rules MUST be referenced, pinned, and replayed as policy artifacts.
- This Constitution does NOT require a specific transparency-log implementation, cryptographic algorithm, or hosting model; it defines required predicates and proof interfaces.
- This Constitution does NOT require disclosure of full clinical charts to verifiers; it requires support for minimal-disclosure verification (subset/redacted representations) sufficient for replay verification under explicit access policy.
- This Constitution does NOT eliminate human review; it defines when automation is admissible and when systems MUST fail-closed into HOLD/manual review.
- This Constitution is NOT a license offer, certification, or regulatory determination.

2.3 Actors (non-limiting)

- Issuer: an entity that constructs and signs canonical receipts (e.g., provider, payer, RCM vendor, clearinghouse, gate operator).
- Verifier: an independent relying party that performs replay verification and predicate checks (e.g., payer adjudication, auditor, regulator).
- Policy Artifact Registry (PAR) operator: an entity that publishes versioned policy programs and artifacts resolvable by verifiers.

- Continuity-Verified Log (CVL) operator: an entity that provides append-only anchoring with inclusion and continuity proofs under freshness bounds.
- Monitor/Witness (optional but recommended): an entity that observes CVL heads to detect equivocation or rollback (split-view).

3. Normative keywords

Normative requirements in this Constitution are indicated using RFC 2119 / RFC 8174 keywords (MUST, SHOULD, MAY, etc.). Text that does not use these keywords is informative unless explicitly labeled otherwise.

4. Definitions

Term	Definition
Receipt	A canonical, machine-verifiable evidence object representing a discrete decision, binding, gate outcome, metric computation, or correction.
Canonicalization	A deterministic serialization procedure that yields stable bytes for hashing and signing, independent of platform, locale, or implementation.
Predicate	A machine-executable condition that MUST hold for admissibility (e.g., signature validity, freshness, policy version/hash match, identity binding, prerequisite satisfaction).
Replay verification	Independent recomputation of referenced structures (graphs or logically equivalent), metrics, predicates, and policy evaluation to reproduce the decision or reproduce failure conditions.
Policy program	A versioned, hash-pinned set of rules and thresholds used to interpret evidence and produce decisions.
Policy drift	Any mismatch between a policy program identifier/version/digest referenced in a receipt and a trusted policy artifact resolved by a verifier.
PAR	Policy Artifact Registry: a resolver for trusted policy artifacts (policy_program_id, version, digest) used during replay verification.

PARR	Policy Artifact Requirements Registry: a resolver for trusted policy-associated requirements artifacts used during replay verification.
CVL	Continuity-Verified Log: an append-only transparency log that publishes signed heads and returns inclusion and continuity proofs under freshness bounds.
Fail-closed	A safety property: when required predicates fail, the system transitions to HOLD/manual review (or deterministic DENY/PEND) rather than proceeding with automation.
Rails	Distinct evidence tracks that produce receipts for different workflow layers (governance, authorization, coding coherence, billing gates, correction).
TOCTOU latch	Time-of-check/time-of-use defense binding a claim (or decision) to an evidence cutover time and a CVL head satisfying freshness, to detect post-hoc edits or late-added evidence.
Conformance Result	A signed validator output summarizing predicate results, badge profile identifiers, and deterministic reason codes for failures.
Conformance badge	A named profile defining required predicates, minimum receipt sets, and reason-code semantics for a workflow or integration rail.

Additional abbreviations (informative): PAMR = Prior Authorization Metrics Receipt.
CPP = Correction Propagation Policy (used with CR for forward-linked corrections/recoupment).

5. Safety property: Fail-Closed Admissibility

5.1 Admissible Automation Rule (normative)

A payment, denial, auto-adjudication, submission, or other financially consequential action is admissible only when replay verification succeeds from machine-verifiable receipts under configured predicates. If it is not replayable evidence, it is not admissible automation.

5.2 Fail-closed default (normative)

When any required predicate fails—including integrity, freshness/continuity, policy drift, missing prerequisites, missing governance evidence, missing required receipts, or identity binding mismatch—the system MUST fail-closed into HOLD/manual review (or deterministic DENY/PEND where policy defines) and MUST emit structured reason codes.

6. Architectural model: five rails and evidence spine

AAS is designed to be realized as an evidence spine composed of multiple rails. Each rail emits canonical receipts that can be independently verified and replayed.

The baseline model uses five rails (non-limiting):

Rail	Gate	Primary receipt(s)	Core admissibility predicate(s) (examples)
1. Governance / HTI	Permit-before-action	HTI Safety Receipt (HTI_SR)	Valid signature; policy pinned; CVL freshness; permit outcome allows downstream action.
2. A-Prime	Bind authorization → claim	Authorization Truth Receipt (ATR) + Claim Truth Receipt (CTR)	Policy drift guard passes; binding key/commitment matches; guardrails satisfied; CVL predicates satisfied.
3. RCM Truth Rail	Episode → codes coherence	Revenue Cycle Truth Receipt (RCTR)	Evidence subgraphs reconstructable; metrics recompute; policy re-evaluates; deterministic exception mapping.
4. HTI2 Billing Gate	Permit-before-bill	Billing Gate Receipt (BGR)	Required upstream receipts present (e.g., HTI_SR); AI-governance preconditions satisfied; anti-circumvention rules enforced.
5. Correction	Correct without rewriting history	Appeal/Overturn Receipt (AOR) + Clawback Receipt (CR)	Forward-links valid; authority present; scope within policy; correction semantics applied during verification.

Multi-rail admissibility (normative): If a workflow requires receipts from multiple rails, automation is admissible only when ALL required receipts are present and ALL required predicates pass for each receipt under a shared freshness policy.

7. Normative invariants

1. I1. Canonical bytes — For every receipt, all verifiers MUST be able to recompute identical canonical bytes from the same receipt content.
2. I2. Signature soundness — A receipt is admissible only if its signature verifies against canonical bytes and the issuer identity is authenticated.
3. I3. Policy pinning — Every decision-bearing receipt MUST reference policy_program_id(s) with version AND digest (hash). Policy drift MUST trigger fail-closed behavior.
4. I4. Continuity & freshness — Every receipt used for admissible automation MUST be anchored to a CVL head satisfying freshness and append-only continuity predicates. Failure MUST trigger fail-closed behavior.
5. I5. TOCTOU defense — Binding receipts MUST support time-of-submission latching to detect post-hoc evidence edits (evidence_cutover_time + cvl_head_id + freshness_window_id).
6. I6. Minimal disclosure — Verification SHOULD minimize PHI disclosure and MUST support subset/redacted representations sufficient for replay verification under explicit access policy.
7. I7. Correction without rewriting — Corrections MUST be forward-linked receipts (AOR/CR). Prior receipts are immutable historical artifacts.
8. I8. Deterministic failure semantics — All failure conditions MUST map to deterministic reason codes and recommended actions suitable for audit and procurement scoring.
9. I9. Auditability — Gate decisions and verification outcomes MUST be recorded in an audit plane (e.g., AuditEvent) linked to receipt identifiers and reason codes.
10. I10. Anti-circumvention (when configured) — Where policy requires AI-use/provenance markers, missing or invalid markers MUST be treated as a failure that routes through governance preconditions rather than bypassing them.

8. Receipt requirements (canonicalization, signatures, policy pinning)

8.1 Required receipt header fields (normative minimum)

Every receipt used for admissible automation MUST include at minimum:

- receipt_type; receipt_version; issuer_id; issued_at
- subject_id (pseudonymous) and one or more contextual identifiers (episode_id and/or authorization_request_id and/or claim_id)
- policy_program_ids[] including {id, version, digest} for each policy program used or asserted
- integrity block including: signature material + CVL anchoring references + upstream receipt references (when applicable)

8.2 Canonicalization (normative)

Implementations MUST define a deterministic canonicalization procedure for receipt payloads prior to hashing/signing. At minimum this includes:

- Deterministic field ordering (schema order or lexicographic) and deterministic ordering of collections that are semantically sets.
- Locale-invariant normalization for numbers, timestamps (ISO 8601 with timezone), and identifiers.
- Explicit treatment of absent vs null vs empty values.
- Digest computation over canonical bytes only (never over presentation formats).

Recommended (informative): For JSON receipts, RFC 8785 (JSON Canonicalization Scheme) is a widely used baseline; however, any canonicalization profile is acceptable if it is fully specified, versioned, and testable via published vectors.

8.3 Signatures and issuer identity (normative)

Receipts MUST be signed over canonical bytes. The signature envelope MUST bind the issuer_id to a cryptographic identity (e.g., certificate chain or equivalent). Key rotation and compromise handling MUST be documented; compromised keys MUST be revocable and revocation MUST be checkable by verifiers.

8.4 Policy program pinning + Policy Artifact Registry (PAR) (normative)

Policy identifiers are first-class evidence. A receipt that omits policy version or digest is non-admissible for automation. During replay verification, verifiers MUST resolve trusted policy artifacts via a PAR (or equivalent trust store) and MUST treat any mismatch or unavailability as a deterministic failure (see reason codes).

8.5 Receipt-type minimum requirements (normative baseline)

The following table summarizes minimum receipt responsibilities. Implementations MAY add fields, but MUST NOT remove required fields without a receipt_version major increment.

Receipt	Purpose	Minimum additional fields (beyond header)	Typical fail-closed triggers (examples)

HTI_SR	Govern AI-assisted DSI episodes (permit-before-action).	dsi_episode_id;
		dsi_engine_id/version/config;
		permit_outcome;
		guardrails[];
		governance policy refs.
			PERMIT_CONDITION_FAILURE;
			STALE_ANCHORING;
			INVALID_SIGNATURE.

ATR	Make PA decision	requested_service_spe	POLICY_DRIFT_GUARD_FAILURE;

	replayable and policy-pinned.	c; evidence_subgraph_refs[]; outcome; guardrails[]; decision_provenance; decision_rationale (reason codes).	EVIDENCE_INSUFFICIENT; INVALID_SIGNATURE.
CTR	Bind claim line items to ATR(s) and enforce TOCTOU latch.	binding_refs to ATR(s); per-code evidence refs; per-code metrics; exception_category + recommended_action_code; time-of-submission latch.	IDENTITY_BINDING_MISMATCH; TOCTOU_LATCH_MISMATCH; CONTINUITY_VERIFICATION_FAILURE.
RCTR	Bind episode→codes coherence metrics for replay by payer/auditor.	clinical episode identifiers; per-code evidence subgraphs; coherence metrics; policy outputs summary.	REPLAY_METRIC_MISMATCH; EVIDENCE_INSUFFICIENT; VERIFICATION_PATH_UNAVAILABLE.
BGR	Permit-before-bill gating for AI-influenced billing events.	ai_influence_indicator (or policy-based inference); upstream receipt refs; gate disposition; failed_preconditions list.	AI_GOVERNANCE_PRECONDITION_FAILURE; MISSING_REQUIRED_RECEIPT; STALE_ANCHORING.
PAMR	Replay-verifiable PA metrics aggregate for reporting/admissibility predicates.	window; slice identifiers; aggregate metrics; computation commitments; policy refs.	EVIDENCE_INSUFFICIENT; POLICY_DRIFT_GUARD_FAILURE.
AOR	Appeal/overturn without rewriting history.	forward-links to ATR/CTR; overturn outcome; reason codes; evidentiary basis; remediation directives.	CORRECTION_AUTHORITY_MISSING; OUT_OF_POLICY_SCOPE.
CR	Clawback/recoup/correct with bounded propagation.	forward-links; scope; nullification semantics; evidentiary requirements; approval authorities; propagation targets.	CORRECTION_AUTHORITY_MISSING; OUT_OF_POLICY_SCOPE; VERIFICATION_PATH_UNAVAILABLE.

8.6 Precondition Failure Record (PFR) (normative recommendation)

When a gate prevents automation or submission due to predicate failure, the system SHOULD emit a machine-interpretable Precondition Failure Record containing at minimum: (a) recommended_action_code, (b) failed_preconditions list, and (c) reason_code_ids. This record is consumable by routing, audit, and remediation workflows.

9. Continuity-Verified Log (CVL) requirements

9.1 Definition (normative)

A CVL is an append-only transparency service that publishes signed heads and returns proofs that allow relying parties to verify:

- Inclusion: a receipt digest is included in the log under a given signed head.
- Continuity: the log evolved append-only from a previous head to a current head (append-only evolution (consistency) proof).
- Freshness: the signed head and inclusion proof satisfy a configured freshness bound (e.g., maximum merge delay / staleness interval).

9.2 Required CVL predicates (normative baseline)

Predicate	What verifier checks	On failure (default)
CVL.FRESHNESS	head_timestamp within freshness window for the workflow/policy.	Fail-closed HOLD; reason STALE_ANCHORING.
CVL.INCLUSION	inclusion proof validates canonical receipt digest under signed head.	Fail-closed HOLD; reason CONTINUITY_VERIFICATION_FAILURE.
CVL.CONSISTENCY	if head advanced, an append-only evolution (consistency) proof demonstrates append-only evolution from a prior signed head to a current signed head.	Fail-closed HOLD; reason CONTINUITY_VERIFICATION_FAILURE.
CVL.HEAD_RECONCILIATION	heads reconcile across	Fail-closed HOLD; reason LOG_EQUIVOCATION_SUSPECTED.

	replicas/participants (gossip/monitoring) as configured.	

9.3 Anti-equivocation (normative recommendation)

CVL operators SHOULD support head reconciliation (gossip/monitoring) and MAY support witness/monitor mechanisms to reduce split-view risk. When CVL.HEAD_RECONCILIATION fails (i.e., equivocation is suspected), verifiers MUST fail-closed and emit reason codes.

9.4 Availability expectations (informative)

Because CVL unavailability can halt admissible automation, deployments SHOULD define SLOs, caching behavior for proofs, redundancy strategy, and explicit fallback semantics (which still MUST remain fail-closed for automation).

10. Replay verification algorithm (normative)

Given a claim (or claim line item), a verifier MUST execute the following steps in order. If any required step fails, the verifier MUST fail-closed and return deterministic reason codes.

10.1 Procedure (normative):

a. Verify sampling specification + selection proof (if sampling is used).

b. Fetch all referenced receipts required by the workflow (HTI_SR, ATR, CTR, RCTR, BGR, and any AOR/CR).

c. Verify receipt signatures over canonical bytes and authenticate issuer identities (including revocation checks).

d. Verify CVL predicates for each receipt: freshness, inclusion, and (when applicable) continuity/consistency; reject stale or unverifiable anchoring.

e. Confirm identity binding across receipts (subject/encounter/window/service type) and binding keys/commitments where defined.

f. Apply forward-linked correction semantics: if AOR/CR exists, treat superseded receipts as non-admissible for automation per correction rules.

g. Reconstruct required structures (Clinical Episode Graph and Evidence Subgraphs or logically equivalent structures) from minimal necessary data under access policy.

h. Recompute coherence metrics and prerequisite predicates using the verifier’s recomputation rules.

i. Resolve policy artifacts via PAR; re-evaluate payer policy programs using pinned versions/digests; confirm policy drift guard passes.

j. Emit an admissibility decision and recommended action code. If all predicates pass → admissible automation (e.g., AUTO_PAY). If any predicate fails → fail-closed HOLD/manual review (or deterministic DENY/PEND as policy defines).

10.2 Verifier output object (normative recommendation)

Verifiers SHOULD produce a structured output suitable for audit and automation routing. At minimum:

- admissibility: {ADMISSIBLE | NOT_ADMISSIBLE}
- recommended_action_code: {AUTO_PAY | HOLD | PARTIAL_PAY | DENY | ROUTE_FOR_INTEGRITY_REVIEW | REQUEST_INFO} (non-limiting)
- reason_code_ids (deterministic failures, if any)
- predicate_results (including per-rail predicate pass/fail status)
- replay_summary including policy identifiers + versions/digests used during replay

10.3 Reference pseudocode (informative)

```
function VERIFY_ADMISSIBILITY(claim, workflow_profile):
  receipts = RESOLVE_REQUIRED_RECEIPTS(claim, workflow_profile)
  if receipts.missing: return FAIL("MISSING_REQUIRED_RECEIPT")

  for r in receipts:
    if !VERIFY_CANONICAL_BYTES(r): return FAIL("NON_CANONICAL_ENCODING")
    if !VERIFY_SIGNATURE(r): return FAIL("INVALID_SIGNATURE")
    if !VERIFY_CVL_PREDICATES(r): return FAIL(REASON_FROM_CVL(r))

  if !VERIFY_IDENTITY_BINDING(receipts): return FAIL("IDENTITY_BINDING_MISMATCH")

  receipts = APPLY_FORWARD_LINKED_CORRECTIONS(receipts)  # AOR/CR

  structures = RECONSTRUCT_STRUCTURES(claim, receipts)   # graphs or equivalent
  if !structures.ok: return FAIL("EVIDENCE_INSUFFICIENT")

  metrics = RECOMPUTE_METRICS(structures, receipts)
  if !metrics.match(receipts): return FAIL("REPLAY_METRIC_MISMATCH")

  policies = RESOLVE_POLICIES_FROM_PAR(receipts)
  if !policies.match(receipts): return FAIL("POLICY_DRIFT_GUARD_FAILURE")
```

```
decision = REEVALUATE_POLICIES(policies, metrics)
return PASS(decision.recommended_action_code)
```

11. Deterministic failure semantics (reason codes)

11.1 Principles (normative)

Reason codes MUST be stable, enumerable, and machine-consumable. They MUST be sufficient to:

- Explain why automation was not admissible without relying on free-text narratives.
- Drive deterministic routing (AUTO_PAY vs HOLD vs DENY/PEND vs integrity review).
- Support procurement scoring, audit trails, regulator review, and root-cause remediation.

11.2 Baseline reason code set (normative baseline)

The baseline set below is REQUIRED for AAS-core interoperability. Implementations MAY define additional codes, but MUST NOT redefine baseline meanings.

Reason code	Category	Meaning (normative)	Default routing (non-limiting)	Typical remediation
INVALID_SIGNATURE	Integrity	Receipt signature fails, issuer identity not authenticated, or revocation check fails.	HOLD; route for integrity review.	Re-issue receipt with valid signature; resolve key/cert; publish revocation status.
NON_CANONICAL_ENCODING	Integrity	Digest mismatch under receipt_version canonicalization rules (non-canonical bytes).	HOLD.	Fix canonicalization; re-emit receipt_version; validate against test vectors.
CONTINUITY_VERIFICATION_FAILURE	Continuity	CVL inclusion/ append-only evolution (consistency) proofs missing	HOLD.	Re-anchor; fetch proofs; repair CVL connectivity; verify append-only

		or invalid.		path.
STALE_ANCHORING	Continuity	Freshness bound violated for required CVL head/proof.	HOLD.	Re-anchor under current head; adjust freshness windows only via policy change.
LOG_EQUIVOCATION_SUSPECTED	Continuity	Divergent signed heads detected (possible split-view/rollback).	HOLD; freeze automation slice as configured.	Head reconciliation/gossip; investigate CVL operator; require witness confirmations.
POLICY_DRIFT_GUARD_FAILURE	Policy	Policy id/version/digest mismatch, or policy artifact unavailable/unverifiable (PAR/PARR failure).	HOLD.	Resolve trusted policy artifact; pin correct version/digest; re-run replay.
MISSING_REQUIRED_RECEIPT	Completeness	A required receipt for the workflow/profile is absent.	HOLD.	Emit missing receipt(s); ensure workflow integration for required rails.
AI_GOVERNANCE_PRECONDITION_FAILURE	Governance	Missing/ invalid governance evidence (e.g., HTI_SR/BGR) when required for AI-influenced workflow.	HOLD; mark noncompliant AI indicator when configured.	Issue/repair governance receipts; enforce anti-circumvention; remediate DSI governance.
PERMIT_CONDITION_FAILURE	Governance	Permit outcome/guardrails not	HOLD or DENY per policy (e.g.,	Satisfy guardrails; obtain

		satisfied for the requested downstream action.	DENY if permit outcome is DENY).	override with authority; re-issue permit receipt.
IDENTITY_BINDING_MISMATCH	Binding	Subject/ episode/service identifiers or binding commitments mismatch across receipts.	HOLD.	Correct identity linkage; re-bind via new receipt; investigate data integrity.
TOCTOU_LATCH_MISMATCH	Integrity	Post-cutover evidence change detected relative to time-of-submission latch.	HOLD.	Issue forward-linked correction (AOR/CR) or re-submit with new latch; audit edits.
EVIDENCE_INSUFFICIENT	Evidence	Prerequisites/ coverage predicates or required documentation not satisfied under policy.	PEND/ REQUEST_INFO or HOLD per policy.	Provide missing documentation; correct evidence graph; re-run evaluation.
EVIDENCE_RESOLUTION_OUT_OF_POLICY	Privacy/ Access	Evidence disclosure blocked by access or data-use policy; verifier cannot resolve required evidence.	HOLD.	Obtain proper authorization; adjust disclosure scope; use redacted/minimal subset.
REPLAY_METRIC_MISMATCH	Replay	Verifier recomputation of metrics/predicates does not match receipt assertions.	HOLD; route for integrity review.	Align metric definitions; fix implementation; publish metric test vectors.
CORRECTION_AUTHORITY_MISSING	Correction	AOR/CR missing	HOLD.	Obtain proper

		required approval authority or attestation.		authority; re-issue correction receipt with required approvals.
OUT_OF_POLICY_SCOPE	Correction	Correction scope violates policy (e.g., out-of-window, wrong slice, forbidden nullification semantics).	HOLD or DENY per policy.	Constrain scope; issue compliant correction; escalate to governance.
VERIFICATION_PATH_UNAVAILABLE	Availability	Required endpoints for proofs/evidence/policy resolution are unavailable within policy time bounds.	HOLD.	Restore availability; use redundancy/caching; re-run verification when path recovers.

11.3 Extensibility (normative)

Implementations MAY define additional reason codes. Additional codes MUST NOT change baseline meanings. Recommended (informative): use a structured namespace (e.g., VENDOR_XYZ:CODE) for non-baseline extensions.

12. Correction without rewriting history (AOR / CR)

12.1 Forward-linked correction (normative)

Appeal/Overturn Receipts (AOR) and Clawback Receipts (CR) MUST forward-link to affected receipts and/or claims. Verifiers MUST apply correction semantics by supersession rules; prior receipts MUST remain immutable historical artifacts.

12.2 Correction semantics (normative baseline)

- AOR MUST encode: {overturn_outcome, overturn_reason_code(s), evidentiary_basis, remediation_directives} and MUST identify which prior decision(s) are superseded.

- CR MUST encode: {scope, nullification semantics, evidentiary requirements, approval authorities, propagation targets} and MUST specify how downstream systems should reconcile.
- If a verifier observes an applicable AOR/CR, it MUST treat superseded receipts as not admissible for automation for the affected scope until corrected state is replay-verifiable.

12.3 Audit and accountability (normative recommendation)

Correction receipts SHOULD include materials_reviewed commitments and authority attestations, and correction events SHOULD be recorded in an audit plane (e.g., AuditEvent) with deterministic reason codes.

13. Conformance artifacts (validator, badge profiles, test vectors)

13.1 Conformance validator + Conformance Result (normative recommendation)

A reference validator (or equivalent) MAY consume canonical receipts and emit a signed Conformance Result that includes:

- badge_profile_id (e.g., CB-1) and badge_profile_version
- validator_id + validator_version + validator_digest
- required_rails and required_predicates (expanded list)
- predicate_results (pass/fail per predicate, per rail)
- replay_verification_summary (what was recomputed and matched)
- recommended_action_code and reason_code_ids

Note (non-limiting): In CB-1 contexts, recommended_action_code MAY be represented as or mapped to fail_closed_action for fail-closed routing (e.g., HOLD/DENY/PEND), without limiting implementations.

13.2 Badge profiles (normative bridge)

A conformance badge profile defines the exact minimum receipt set per workflow and the required predicates per rail. Badge profiles are the clean bridge from “whitepaper” → “standard people implement.”

Badge profile	Scope	Notes
CB-1 Permit-to-Pay	Baseline admissibility for permit-before-action → authorization → claim → billing gate → correction.	Defines required predicates per rail, minimum receipt sets per workflow, and baseline reason-code semantics.

13.3 Minimum open conformance pack (normative recommendation; reference-only acceptable)

To make AAS implementable and independently testable, issuers and validators SHOULD publish a tiny conformance pack containing:

- Canonicalization test vectors (input receipt → canonical bytes → digest) for each receipt_version.
- A small set of sample receipts (HTI_SR, ATR, CTR, RCTR, BGR, AOR/CR) including both passing and failing examples.
- A verifier script (reference-only acceptable) that checks canonicalization, signatures, CVL proofs, and replay verification scaffolding.
- Adversarial/tamper cases: modified fields, stale heads, policy drift, TOCTOU latch mismatch.
  Example pack (this Zenodo record): CTC_CB-1_Open_Conformance_Pack_v1.1.0_REALVALUES_PLUSPLUS.zip.

13.4 Conformance claims (normative)

Any public claim of conformance to AAS (or a badge profile) MUST specify the badge_profile_id and badge_profile_version, and validator_version and validator_digest used to produce the Conformance Result.

14. FHIR interoperability constitution (DocumentReference + Provenance + AuditEvent)

14.1 Interoperability pattern (normative recommendation)

For interoperability, each canonical receipt SHOULD be transported and indexed as a FHIR DocumentReference (payload = canonical receipt bytes). A corresponding FHIR Provenance resource SHOULD capture who/what/when and SHOULD link upstream receipts via Provenance.entity and/or DocumentReference.relatesTo. When represented in FHIR, security-relevant gate decisions and verification outcomes MUST be recorded as FHIR AuditEvent instances referencing the associated receipt(s) and claim context.

14.2 Minimal mapping (informative)

CTC concept	FHIR resource(s)	Notes
Receipt payload bytes	DocumentReference.content (+ Binary)	DocumentReference indexes the receipt; Binary can carry raw bytes.
Who/what/when + signature	Provenance (+ Provenance.signature)	Bind issuer identity and signature to receipt targets.
Gate decision / verification outcome	AuditEvent	Record HOLD vs AUTO_PAY decisions with reason codes.
Upstream/forward links	DocumentReference.relatesTo; Provenance.entity	Represent source/derivation/revision relationships.

CVL proofs / freshness	Extension or separate DocumentReference (LogReceipt)	FHIR core does not define Merkle proofs; use explicit extensions.

14.3 Interop invariants (normative)

- FHIR envelopes MUST preserve the canonical receipt bytes exactly; verifiers MUST hash/sign/verify the canonical bytes, not rendered text.
- If Merkle proofs, freshness windows, or log heads are carried, they MUST be represented with clearly named extensions or separate LogReceipt objects so verifiers can validate them without ambiguity.
- When using FHIR AuditEvent, records MUST reference the receipt identifiers and include deterministic reason codes when automation is not admissible.

15. Security considerations

AAS is security-sensitive. Implementations SHOULD explicitly document their threat model, trust boundaries, and operational controls. The following are non-limiting considerations:

- Key management: protect signing keys (e.g., HSM/secure enclave), support rotation, and publish revocation status checkable by verifiers.
- Receipt tampering: signatures MUST be over canonical bytes; verifiers MUST reject non-canonical encodings.
- Log equivocation (split-view): implement head reconciliation, monitoring/witnesses where feasible; fail-closed on suspected equivocation.
- Stale anchoring: enforce freshness bounds as policy; do not treat stale proofs as “close enough.”
- Policy substitution/drift: enforce policy id+version+digest pinning; treat PAR unavailability as fail-closed for automation.
- TOCTOU and post-hoc evidence edits: use time-of-submission latch; require forward-linked correction receipts rather than silent mutation.
- DoS and availability: design for redundancy and caching of proofs; nevertheless, automation MUST remain fail-closed under missing proofs.

16. Privacy considerations

AAS is designed to be verifiable while minimizing disclosure. Implementations MUST treat PHI minimization as a first-class constraint.

- Pseudonymous identifiers: receipts SHOULD avoid direct patient identifiers; use pseudonymous subject/episode identifiers where possible.
- Minimal disclosure verification: verifiers SHOULD request only the subset of evidence needed to reconstruct referenced subgraphs/predicates.
- Policy-bounded evidence resolution: evidence resolution SHOULD be gated by explicit access and data-use policies; failures must emit reason codes.
- Auditability: accesses and disclosures SHOULD be logged (e.g., AuditEvent) with traceable purpose-of-use and policy references.

17. Governance, change control, and registries

17.1 Versioning (normative)

Changes to canonicalization rules, required fields, reason code meanings, or policy interpretation semantics are security-sensitive and MUST be versioned. Silent changes are non-admissible.

Recommended (informative): use semantic versioning:

- MAJOR: breaking change (canonicalization bytes, required predicates, required fields, or reason code meaning changes).
- MINOR: backward-compatible additions (new optional fields, new optional predicates, additional reason codes).
- PATCH: editorial fixes that do not affect canonical bytes or predicate semantics.

17.2 Registries (normative recommendation)

To support ecosystem adoption, AAS SHOULD maintain public registries for:

- Receipt types and receipt_version identifiers.
- Reason codes (baseline + extension namespaces).
- Badge profile identifiers and versions (e.g., CB-1).
- Canonicalization profile identifiers (especially when multiple profiles are supported).

17.3 Deprecation and migration (normative)

When a breaking change occurs, implementations MUST either: (a) support parallel verification of prior versions, or (b) publish explicit migration guidance and, when appropriate, migration receipts. Backwards compatibility MUST be explicit; implicit behavior changes are not admissible.

Appendix A. Minimal receipt schema checklist (informative)

The following checklist is a quick implementation guide. It is informative, but reflects normative requirements stated above.

1. Define receipt_type and receipt_version and publish schema.
2. Define canonicalization profile and publish canonicalization test vectors.
3. Sign canonical bytes; publish issuer identity and revocation strategy.
4. Anchor receipt digests to a CVL and publish how inclusion/consistency/freshness proofs are obtained.
5. Pin policy artifacts (id, version, digest) and publish PAR resolution rules.
6. Implement TOCTOU latch for claim/decision binding when post-hoc edits are a risk.
7. Emit deterministic reason codes and recommended action codes on failures.
8. Support forward-linked corrections (AOR/CR) without rewriting history.

Appendix B. Minimal FHIR bundle patterns (informative)

A minimal interoperable pattern is: DocumentReference (receipt bytes) + Provenance (issuance/signature/upstream links) + AuditEvent (gate/verification outcomes). Concrete bundle examples should be published alongside the conformance pack.

Appendix C. Example Conformance Result (informative)

The following is a non-normative example shape for a Conformance Result object:

```
{
  "badge_profile_id": "CB-1",
  "badge_profile_version": "1.1",
  "validator_id": "urn:ctc:validator:reference",
  "validator_version": "1.0.0",
  "validator_digest": "sha256:…",
  "evaluated_at": "2026-01-02T00:00:00Z",
  "outcome": "FAIL",
  "fail_closed_action": "HOLD",
  "reason_code_ids": ["POLICY_DRIFT_GUARD_FAILURE", "STALE_ANCHORING"],
  "predicate_results": {
    "CANON_OK": true,
    "SIG_OK": true,
    "POLICY_PIN_OK": false,
    "CVL_OK": false,
    "ID_BIND_OK": true
  }
}
```

Appendix D. Reason code registry (normative baseline)

The baseline reason code list and meanings in §11 are normative. Implementations MUST preserve these meanings across versions.

Appendix E. Canonicalization profile guidance (informative)

Canonicalization is a security boundary. Implementations should treat canonicalization profiles as cryptographic protocol material:

- Version canonicalization profiles explicitly.
- Publish cross-language test vectors (Python/Go/Java/etc.).
- Include adversarial vectors (reordered fields, whitespace changes, numeric formatting, timezone normalization).
- Define how to canonicalize nested objects, arrays, and map-like collections.

Appendix F. References (informative)

These references are provided for reviewer convenience and implementer alignment. Normative requirements are defined in the main body of this Constitution.

- RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. https://www.rfc-editor.org/rfc/rfc2119
- RFC 8174: Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. https://www.rfc-editor.org/rfc/rfc8174
- RFC 8785: JSON Canonicalization Scheme (JCS). https://www.rfc-editor.org/rfc/rfc8785
- RFC 6962: Certificate Transparency. https://www.rfc-editor.org/rfc/rfc6962
- IETF SCITT Working Group (architecture and drafts). https://datatracker.ietf.org/wg/scitt/
- Sigstore Rekor transparency log documentation. https://docs.sigstore.dev/logging/overview/
- SLSA Specification v1.0. https://slsa.dev/spec/v1.0/
- in-toto Attestation Framework spec (Statement v1). https://github.com/in-toto/attestation/blob/main/spec/v1/statement.md
- CMS Interoperability and Prior Authorization Final Rule (CMS-0057-F). https://www.cms.gov/cms-interoperability-and-prior-authorization-final-rule-cms-0057-f

- ASTP/ONC Health Data, Technology, and Interoperability (HTI-1) Final Rule resources. https://www.healthit.gov/topic/laws-regulation-and-policy/health-data-technology-and-interoperability-certification-program

- HL7 FHIR R4 (v4.0.1): Provenance https://hl7.org/fhir/R4/provenance.html ; AuditEvent https://hl7.org/fhir/r4/auditevent.html ; DocumentReference https://hl7.org/fhir/R4/documentreference.html

- W3C PROV-DM: The PROV Data Model. https://www.w3.org/TR/prov-dm/

- NIST SP 800-30 Rev.1: Guide for Conducting Risk Assessments. https://csrc.nist.gov/pubs/sp/800/30/r1/final

- OWASP Threat Modeling Process (STRIDE). https://owasp.org/www-community/Threat_Modeling_Process

- LINDDUN Privacy Threat Modeling framework. https://linddun.org/