Network Working Group
Internet-Draft
Intended status: Informational
Expires: 2 December 2026

J. A. Gomes Marques
Asqav

31 May 2026

Compliance Profile of Signed Action Receipts for AI Agents
draft-marques-asqav-compliance-receipts-05

Abstract

This document defines a multi-jurisdiction compliance profile of the signed action receipt format used by AI agents to record machine-readable evidence of access-control decisions. The profile binds receipt fields to two regulatory surfaces: on the European Union side, Articles 12 and 26 of the EU AI Act (Regulation (EU) 2024/1689) and Article 17 of DORA (Regulation (EU) 2022/2554); on the United States side, the NIST AI Risk Management Framework, the Colorado AI Act, the Texas Responsible AI Governance Act, the New York Department of Financial Services Cybersecurity Regulation (23 NYCRR Part 500), the HIPAA Security Rule, SEC Rule 17a-4, and the Cyber Incident Reporting for Critical Infrastructure Act of 2022 (CIRCIA). Working entirely within the existing wire format, canonicalization transformation, and signing algorithms of the underlying receipt format, the profile tightens a subset of the OPTIONAL fields to REQUIRED, imposes a retention floor, and requires at least one timestamping anchor (RFC 3161 or OpenTimestamps). It registers OPTIONAL extension fields for risk and incident classification, cross-agent envelope binding, per-action freshness and integrity, build provenance, threat-framework taxonomy, and server-built enforcement attestation, each subject to false-attestation guards where applicable, and registers receipt type namespaces for passive-telemetry and result-bound observation receipts. The full field set and its normative requirements are defined in the body of this document.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

1.1. Profile, Not Fork

[ACTA-RECEIPTS] specifies a generic, signed receipt envelope for recording machine-to-machine access control decisions made by AI agents. Section 2.2 of [ACTA-RECEIPTS] defines a common payload field set in which all fields except type, issued_at, and issuer_id are OPTIONAL. Section 5.7 of [ACTA-RECEIPTS] introduces hash chaining (previousReceiptHash) inside an optional Commitment Mode extension. [ACTA-RECEIPTS] does not define receipt retention, does not require timestamping anchors, and does not bind to any regulatory regime.

This document is an additive overlay on [ACTA-RECEIPTS]: it constrains fields the upstream draft leaves OPTIONAL, fixes their values where regulation requires, and registers a set of extension fields with reserved names spanning regulatory classification, cross-agent envelope binding, per-action freshness and integrity, build provenance, threat-framework taxonomy, and server-built enforcement attestation. The full extension-field set is defined in Section 5.5 and the sections that follow it. A Compliance Receipt remains a conformant [ACTA-RECEIPTS] receipt. Field references use upstream field names rather than section numbers, to reduce maintenance hazard if upstream re-numbers in a future revision.

1.2. Scope

This document fills the regulatory binding gap on two surfaces. Section 5 binds the receipt to European Union obligations: Article 12 (record-keeping) and Article 26 (deployer obligations) of the EU AI Act, and Article 17 (ICT-related incident management) of DORA. Section 6 binds the receipt to United States obligations: the voluntary functions of the NIST AI Risk Management Framework, the deployer obligations of the Colorado AI Act and the Texas Responsible AI Governance Act, the audit-trail and incident-reporting obligations of NYDFS Part 500, the audit controls and documentation retention of the HIPAA Security Rule, the broker-dealer recordkeeping requirements of SEC Rule 17a-4, and the covered-incident reporting requirements of CIRCIA.

The bindings are written from the Deployer’s perspective, where Deployer is used in the regime-specific sense (Article 3(4) of [EU-AI-ACT] for EU bindings; Section 6-1-1701(6) of the Colorado Revised Statutes for Colorado bindings). Where another statute uses a different term (Provider, Financial Entity, Covered Entity for HIPAA, Covered Entity for NYDFS, Broker-Dealer for SEC, Covered Entity for CIRCIA), the binding section names the term as the source statute uses it.

A verifier that implements only [ACTA-RECEIPTS] can cryptographically validate a profile receipt but cannot attest the additional compliance bindings of this document.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document.

Action: An operation performed by an AI agent that is subject to a policy evaluation. Examples include a tool invocation, an external API call, a write to durable storage, and the issuance of an irreversible instruction to another system.

Action Receipt: A signed envelope conforming to [ACTA-RECEIPTS] that records the policy evaluation result for a single Action.

Compliance Receipt: An Action Receipt that additionally satisfies the requirements of this profile.

Deployer (EU AI Act): As defined in Article 3(4) of [EU-AI-ACT].

Deployer (Colorado AI Act): As defined in Section 6-1-1701(6) of the Colorado Revised Statutes, as enacted by [COLORADO-AI-ACT].

High-Risk AI System (EU AI Act): As defined in Article 6 of [EU-AI-ACT].

High-Risk AI System (Colorado AI Act): As defined in Section 6-1-1701(9) of the Colorado Revised Statutes, as enacted by [COLORADO-AI-ACT].

Financial Entity: As defined in Article 2(2) of [DORA], for entities listed in Article 2(1).

Covered Entity (HIPAA): As defined in 45 CFR 160.103, namely a health plan, a health care clearinghouse, or a health care provider that transmits health information in electronic form in connection with a covered transaction.

Covered Entity (NYDFS): As defined in 23 NYCRR 500.1(e), namely any person operating under or required to operate under a license, registration, charter, certificate, permit, accreditation or similar authorization under the Banking Law, the Insurance Law or the Financial Services Law, regardless of whether the covered entity is also regulated by other government agencies.

Broker-Dealer: As defined in section 3(a)(4) and 3(a)(5) of the Securities Exchange Act of 1934, subject to recordkeeping under [SEC-17A-4].

Covered Entity (CIRCIA): As to be defined in the final rule promulgated under the Cyber Incident Reporting for Critical Infrastructure Act of 2022. Pending publication of the final rule, the term is interpreted in accordance with the statutory definition at 6 U.S.C. 681 and CISA’s notice of proposed rulemaking.

Audit Pack: A bundle of Compliance Receipts, the chain commitments that link them, the public verification keys, the trust anchor metadata, and the regime mapping required by Sections 5 and 6 of this document, packaged for delivery to a regulator or auditor.

3. Relationship to ACTA-RECEIPTS

This profile is an additive overlay on [ACTA-RECEIPTS]. It does not modify the envelope, the canonicalization rule, the signature object, or the algorithm registry of [ACTA-RECEIPTS].

The following normative statements apply.

* Implementations of this profile MUST produce receipts that are cryptographically verifiable by a conformant [ACTA-RECEIPTS] verifier under the canonicalization rules (JCS, [RFC8785]) and the signature scope of [ACTA-RECEIPTS] Section 5.6.

* Implementations of this profile MUST NOT introduce new top-level fields in the signed payload that conflict with names reserved by [ACTA-RECEIPTS].

* Implementations of this profile MAY use any signature algorithm permitted by [ACTA-RECEIPTS]: EdDSA (Ed25519, mandatory-to-implement, [RFC8032]), ES256 (ECDSA using P-256 and SHA-256, [RFC7518]), and ML-DSA-65 ([FIPS204]).

* Where [ACTA-RECEIPTS] marks a field OPTIONAL and this profile marks the same field REQUIRED, the stricter requirement applies to Compliance Receipts.

A receipt that fails any MUST clause of this profile is not a Compliance Receipt. It MAY still be a valid [ACTA-RECEIPTS] receipt.

This profile differentiates from [ACTA-RECEIPTS] on three axes: mandatory hash-chain linkage (upstream Commitment Mode is OPTIONAL), mandatory anchoring with RFC 3161 or OpenTimestamps (both RECOMMENDED; upstream lists Sigstore Rekor in its Implementation Status appendix as an OPTIONAL temporal anchor), and a retention floor tied to specific regulatory articles (upstream is silent on retention).

4. Canonicalization Scope

This section is normative. The canonicalization rule itself (JCS, [RFC8785]) is inherited unchanged from [ACTA-RECEIPTS]; this section bounds the inputs the rule is applied to, so that the cross-implementation byte equality on which the hash chain of Section 5.3, the anchor scope of Section 5.4, and the cross-agent binding of Section 5.6 all depend is achievable in practice.

IEEE-754 floating-point numbers MUST NOT appear in the canonical form covered by a SHA-256 digest under this profile. Callers MUST serialize numeric values that are not exact integers in the IEEE-754 safe integer range (the closed interval from minus (2 to the 53 minus 1) to plus (2 to the 53 minus 1) inclusive) either as JSON strings or as integer-rational pairs (numerator and denominator as JSON numbers within that safe integer range) before the canonicalization step.

Rationale: Section 3.2.2.3 of [RFC8785] specifies, by reference to Section 7.1.12.1 of ECMA-262, a byte-stable serialization that in principle covers all IEEE-754 double-precision values, integer or fractional. In practice, several widely deployed JSON serializers do not implement the ECMA-262 Number-to-String algorithm with byte fidelity: Python json.dumps, Go encoding/json, and Java Jackson (without explicit configuration) all produce different byte sequences from the same IEEE-754 double in documented cases (round-to-even ties, subnormal values, large-magnitude values requiring scientific notation). Compliance and regulatory contexts (monetary amounts, retention thresholds, anchoring intervals) additionally prefer exact integer or string-encoded decimal representations because float rounding loses the exact bytes that auditors quote. This profile therefore shifts the canonicalization burden off implementers (who would otherwise have to verify ECMA-262 conformance of an underlying JSON library) and onto callers, who are in a better position to choose a portable representation for the use case at hand. A receipt that carries a floating-point number in a digest-covered field is not guaranteed to verify across implementations even when each implementation independently conforms to [RFC8785], because the conformance burden has not been met by every mainstream JSON library.

Tool-version-specific semantic equivalence is OUT OF SCOPE for the chain layer of this profile. The chain layer guarantees byte equality only. Examples of semantic equivalence that this profile does not assert and does not require a verifier to assert: SQL keyword case folding (SELECT vs select), filesystem path normalization (trailing slash, redundant separators, symlink resolution), Unicode normalization in any form (NFC, NFD, NFKC, NFKD); Section 3.1 of [RFC8785] requires that all components depending on JCS preserve Unicode string data as-is, and Section 3.2.2.2 of [RFC8785] serializes each code point without normalization, so callers MUST NOT rely on a verifier normalizing strings before comparison, locale-aware string collation (Turkish dotted-i, German sharp-s case folding, ICU collation tables), numeric tolerance (1.0 vs 1, 1e3 vs 1000), or URL percent-encoding choices below the RFC 3986 unreserved set. Higher-level semantic equivalence is a per-tool concern and, where required by a regulator, MUST be expressed in the policy artefact resolved through policy_digest (Section 5.2.2) rather than in the chain.

The chain layer of this profile answers a single question for a verifier or a regulator: did the same canonicalized bytes pass through agent X at wall-clock time T, as fixed by the anchor evidence of Section 5.4. Anything beyond that question, including whether two byte sequences are semantically equivalent under a downstream tool, whether a policy update materially changed the meaning of a previously accepted Action, or whether a counterparty’s

interpretation of the same bytes matched the originator's, is the verifier's concern and is supported by the Audit Pack manifest (Section 8) and the verifier reporting fields of Section 9.3, not by the chain itself.

5. Receipt Field Profile

This section enumerates fields defined by [ACTA-RECEIPTS] and states the additional requirements that this profile places on them. Field names follow [ACTA-RECEIPTS] exactly.

Compliance Receipts MUST use the upstream wire field name signature for the signature object, exactly as defined in Sections 2.1 and 2.1.1 of [ACTA-RECEIPTS]. The keys inside that object are alg, kid, sig. Implementations whose internal storage uses a different field name MUST translate to signature on emission and on canonicalization for verification; receipts that appear on the wire under any other top-level field name are non-conformant to [ACTA-RECEIPTS] and to this profile. Anchors MUST be projected into a top-level anchors array with a type discriminator and a value field carrying the anchor bytes (base64-encoded for binary payloads). Flat-column implementations MUST project on emission and Audit Pack export.

5.1. Common Payload Fields

5.1.1. type

Compliance Receipts MUST set type to a value drawn from the namespace protectmcp:decision, protectmcp:restraint, or protectmcp:lifecycle, or to an extension namespace registered for use with this profile.

5.1.2. issued_at

REQUIRED upstream and in this profile. The value MUST be an ISO 8601 timestamp with an explicit timezone. The producing system MUST source the value from a clock synchronized to a recognized time authority and MUST NOT backdate the value. Verifiers MUST reject receipts whose issued_at is more than 300 seconds ahead of the verifier's own clock. Verifiers MUST NOT reject a receipt solely because issued_at lies in the past; past skew is bounded by the applicable retention floor in Sections 5 and 6, not by freshness. Historical receipts within retention MUST verify on the same path as fresh ones.

5.1.3. issuer_id

REQUIRED upstream and in this profile. The value MUST identify a legal entity, not a natural person. Where the producing system is operated by a Deployer, the issuer_id MUST resolve, through the trust anchor metadata in the Audit Pack, to a record naming the Deployer. To preserve the upstream Section 2.2 invariant that issuer_id MUST match the kid field of the signature object, Compliance Receipts MUST place the same value in both issuer_id and kid; the verifier resolves that value to a public key through the Audit Pack trust-anchor metadata rather than through the well-known JWK Set endpoint or the RECOMMENDED sb:issuer:<base58-fingerprint> form of [ACTA-RECEIPTS] Section 2.1.1. This profile thereby supersedes the upstream RECOMMENDED kid format for Compliance Receipts; the upstream RECOMMENDED format remains valid for non-Compliance receipts.

Implementations SHOULD use a Legal Entity Identifier (LEI) as defined by [ISO17442] where one is allocated to the Deployer. Examples and test fixtures MUST use a placeholder whose four-character LOU prefix (positions 1-4) is not allocated in the GLEIF Local Operating Unit code list, whose positions 5-6 are the ISO 17442 reserved value 00, and whose two trailing characters (positions 19-20) are the ISO 7064 mod 97-10 check digits computed over positions 1-18 (for example 000000000000000098, where the all-zero 18-character base produces the check digits 98 per the ISO 17442-1:2020 Annex A check-digit algorithm, which converts any letters in positions 1-18 to digits A=10 ... Z=35 before the mod 97-10 computation; for an all-zero base the conversion is a no-op); implementations MUST NOT use a real third-party LEI in documentation or test data. Where no LEI is allocated and the Deployer is a US entity, an Employer Identification Number (EIN) issued by the United States Internal Revenue Service or a Central Index Key (CIK) issued by the United States Securities and Exchange Commission MAY be used, expressed as the bare numeric string. Decentralized Identifiers ([W3C-DID]) MAY be used otherwise. Implementations MUST treat the value as opaque on verification; identifier resolution is out of scope for this profile.

issuer_id values MUST be bare identifiers without a scheme prefix where the scheme is unambiguous from the value’s syntactic form. An LEI is the 20-character alphanumeric string defined by [ISO17442] and is self-identifying through its length and check-digit structure; implementations MUST emit the bare 20-character LEI without a lei: or other scheme prefix. EINs and CIKs are likewise emitted as the bare numeric string. Decentralized Identifiers ([W3C-DID]) carry their own scheme prefix (did:) as defined by the DID specification and that prefix is intrinsic to the identifier syntax rather than an added scheme tag. The same kid-equals-issuer_id invariant requires signature.kid to be the bare identifier in the same form. The worked

example in Appendix "Worked Example (Informative)" uses the bare 20-character placeholder LEI 00000000000000000098; conformant cloud emitters and SDK clients MUST match this form on the wire.

5.1.4. payload_digest (OPTIONAL upstream, REQUIRED in this profile)

REQUIRED for Compliance Receipts. The value MUST follow the upstream object form (hash, size, optional preview) defined in Section 2.2 of [ACTA-RECEIPTS]; this profile does not redefine the wire shape. The associated payload that this digest covers MUST be retained for the period mandated by the most restrictive applicable regime in Sections 5 and 6 of this document. Implementations MUST NOT discard the underlying payload while a receipt that references it is still within its retention window.

5.1.5. action_ref (OPTIONAL upstream, REQUIRED in this profile)

REQUIRED for Compliance Receipts. The value is a SHA-256 hash of the canonical Action representation as defined in [ACTA-RECEIPTS]. This profile uses action_ref as the primary join key for cross-engine reconstruction during an audit.

5.1.6. sandbox_state (OPTIONAL upstream, REQUIRED for High-Risk in this profile)

REQUIRED for receipts produced by High-Risk AI Systems under either [EU-AI-ACT] or [COLORADO-AI-ACT]. Upstream defines sandbox_state as an OS-level containment status and restricts the value to one of enabled, disabled, or unavailable; this profile inherits that enumeration unchanged. A Deployer that operates a High-Risk AI System and produces a stream of receipts in which sandbox_state is consistently disabled SHOULD treat that stream as a finding under the applicable risk-management documentation requirement (Article 9 of [EU-AI-ACT] for the Provider’s risk management system, with which a Deployer operating per Article 26(1) is required to be consistent; Section 6-1-1703(2) of the Colorado Revised Statutes) and document the rationale in the Audit Pack metadata.

5.1.7. iteration_id (OPTIONAL upstream, REQUIRED for multi-step in this profile)

REQUIRED for multi-step agent workflows. The value MUST be stable across all receipts emitted within the same logical task or session so that a regulator can reconstruct the full chain of Actions. iteration_id is distinct from the upstream session_id field defined in [ACTA-RECEIPTS] Section 3.1.1, which is an opaque MCP session identifier. A Compliance Receipt MAY carry both: session_id for MCP-session correlation and iteration_id for logical-task correlation.

5.2. Decision Receipt Fields (type protectmcp:decision)

The decision field value MUST be allow, deny, rate_limit, or observation. Implementations using a different internal vocabulary (e.g. permit for allow) MUST normalise on emission and on Audit Pack export. The observation value records that an Action was observed and the receipt was signed without any policy evaluation having taken place; it is the regulator-honest alternative to emitting allow when no policy matched, and MUST NOT appear in a receipt of type protectmcp:decision. A producing system that has not evaluated a policy for an Action MUST either refuse to issue a Compliance Receipt for that Action or MUST emit the receipt with type protectmcp:lifecycle and decision observation; in the latter case the upstream policy_decision internal field, if present in the producing system's internal vocabulary, takes the literal value none, which the emitter MUST map to observation on the wire. Verifiers MUST reject a Compliance Receipt that carries decision observation together with type protectmcp:decision; conversely, a Compliance Receipt of type protectmcp:lifecycle MAY carry decision observation in addition to the other three vocabulary values. The policy_digest requirement of Section 5.2.2 applies to observation receipts in the form of a digest of the producing system's "no policy matched" sentinel policy artefact, which the Deployer MUST retain alongside its other policy artefacts for the applicable retention window.

The upstream tool_name field (REQUIRED in [ACTA-RECEIPTS] Section 3.1.1) is REQUIRED for Compliance Receipts of type protectmcp:decision.

5.2.1. reason (OPTIONAL upstream, REQUIRED for deny/rate_limit in this profile)

REQUIRED for Compliance Receipts where decision is deny or rate_limit. The value MUST be a machine-readable reason code drawn from a vocabulary documented in the Deployer's Audit Pack metadata.

5.2.2. policy_digest (OPTIONAL upstream, REQUIRED in this profile)

REQUIRED for Compliance Receipts. The value MUST be of the form sha256:<hex> and MUST reference a policy artefact that the Deployer retains for the applicable retention window. Verifiers MUST reject Compliance Receipts whose policy_digest does not resolve in the Audit Pack.

5.2.3. scanner_decisions (OPTIONAL)

An OPTIONAL scanner_decisions field MAY appear in a Compliance Receipt of type protectmcp:decision. When present, its value MUST be a JSON array of objects, where each object records the outcome of one content scanner plug-in that ran during policy evaluation. Each object has the following members:

* scanner_id (string, REQUIRED): stable identifier for the scanner plug-in (e.g. presidio, llm-guard, cedar).

* scanner_version (string, REQUIRED): vendor-reported version string for the scanner instance that produced the outcome.

* scanner_decision (string, REQUIRED): one of allow, scan_blocked, or observation; matches the wire vocabulary of the parent decision field where applicable.

* latency_ms (integer, OPTIONAL): wall-clock time in milliseconds the scanner took to produce the outcome.

* signature (string, OPTIONAL): vendor-supplied detached signature over the scanner outcome, used by Deployers that require non-repudiation per scanner instance.

The field is OPTIONAL because Deployers running a single in-process scanner (the common starter deployment) gain no auditing value from echoing the outcome and would only pay an envelope-size cost. The per-scanner signature member is OPTIONAL for the same reason: cryptographic non-repudiation across multiple independent scanner vendors is an Enterprise-tier concern and not mandated by this profile. Verifiers MUST tolerate the absence of scanner_decisions and MUST NOT infer that no scanners ran from its absence.

5.3. Hash-Chain Linkage (OPTIONAL upstream, REQUIRED in this profile)

Upstream Commitment Mode introduces previousReceiptHash as part of an optional extension. This profile makes the linkage REQUIRED. Implementations MUST emit a previousReceiptHash field, populated per the digest-scope rule of Section 5.7 of [ACTA-RECEIPTS]: the lowercase hex encoding of SHA-256 over the canonical signing-input bytes of the immediately prior receipt emitted by the same issuer_id, where the canonical signing-input bytes are the JCS-canonical serialization ([RFC8785]) of the predecessor’s signed payload object (the same bytes the predecessor’s cryptographic signature covers), NOT the envelope object that additionally includes the signature or anchors top-level keys. The first receipt in a chain MUST set this field to the all-zero SHA-256 value (this profile’s stipulation;

[ACTA-RECEIPTS] Section 5.7 specifies only the digest scope of subsequent links). JSON key is the literal previousReceiptHash (camelCase, case-sensitive); snake_case aliases MUST NOT appear on the wire.

Rationale for the payload-bytes (signing-input) digest scope rather than the envelope-including-signature digest scope: the chain layer's purpose is to make after-the-fact alteration of the predecessor's signed content detectable, and the predecessor's signed content is exactly the bytes its signature covers (the canonical payload object). Digesting those bytes binds the chain to what A actually attested to, is recomputable offline from the predecessor's payload alone, and matches Section 5.7 of [ACTA-RECEIPTS]. The chain does not need to bind the predecessor's signature value directly because the predecessor's signature is verified independently under Section 9.1, and cross-agent envelope integrity (where binding the peer's signature value matters) is the role of counterparty_binding per Section 5.6, which digests at the envelope-including-signature scope precisely because the peer signature is the load-bearing artefact in the cross-agent case. Implementations that previously digested the envelope-including-signature object MUST migrate to the signing-input scope before emitting chained receipts under this profile; verifiers MUST recompute under the signing-input scope when checking previousReceiptHash.

Each issuer MUST maintain a single linear per-agent chain. When one agent identity emits receipts from multiple concurrent execution paths (for example parallel tool calls dispatched within a single agent loop, or fan-out work performed by a thread pool inside one issuer), the issuer MUST serialize emission through a single predecessor pointer at a time: each newly emitted receipt's previousReceiptHash MUST resolve to the SHA-256(JCS(receipt)) of the immediately prior receipt emitted by that same issuer_id, taken in emission order, regardless of which concurrent execution path produced it. Parallel sub-chains within one agent identity (for example, a per-receipt chain_id discriminator that would partition one issuer's stream into multiple independently advancing chains) are NOT defined by this profile. An issuer that requires parallel sub-chains MUST express each parallel path as a distinct agent identity, with its own issuer_id value, its own signing key, and its own per-agent chain rooted at the all-zero SHA-256 genesis value. Rationale: deterministic verification of the chain segment covering an audit window, as required by the regime bindings of Sections 5 and 6 (in particular Section 6.3.2, Section 7.5.1, and Section 7.6.1), depends on a single linear total order over the receipts emitted under each agent identity; a verifier reconstructing the chain from a regulator-supplied issuer_id needs that ordering to be well-defined without out-of-band metadata.

5.4. Anchoring (No Upstream Equivalent)

[ACTA-RECEIPTS] lists Sigstore Rekor in its Implementation Status appendix as an OPTIONAL temporal anchor. This profile imposes a normative anchoring requirement.

Compliance Receipts MUST be anchored. An anchor is an [RFC3161] timestamp token covering the signed envelope, an [OPENTIMESTAMPS] commitment covering the envelope, or both; implementations SHOULD emit both forms. For both anchor types, the bytes committed are SHA-256(JCS(envelope_minus_anchors)), where envelope_minus_anchors is the wire envelope object with the anchors top-level key removed prior to canonicalization, leaving the two-key object {payload, signature}. The anchors key MUST be removed from the object, not set to null or to an empty array; these produce different JCS output and break interoperability (mirroring the upstream Section 5.6 stripping rule). The anchor thereby binds payload and signature without being self-referential. The anchor evidence MUST be retained alongside the receipt for the applicable retention window. Verifiers MUST reject Compliance Receipts that lack at least one valid anchor.

An anchor MAY be attached after issuance if the receipt is persisted with an unambiguous pending marker and the anchor lands within a documented bound. For [OPENTIMESTAMPS], this profile imposes a 7-day deadline; this is a profile-imposed bound, not a property of the OpenTimestamps protocol, whose calendar-to-block upgrade time depends on the calendar operator’s publication interval. [RFC3161] tokens MUST be obtained synchronously. A verifier MUST treat a pending receipt as non-conformant once the bound elapses.

The anchor MAY cover an aggregate of receipts (for example, a Merkle root over a batch) rather than each receipt individually, provided that the inclusion proof linking the receipt to the aggregate is retained alongside the receipt and the aggregate anchor.

Where the anchor type is [RFC3161], the full TimeStampResp DER bytes MUST be retained, sufficient for offline verification by a holder with access to the TSA’s published public key. Time-stamp tokens carrying ESSCertIDv2 per [RFC5816] MUST be accepted by Compliance Verifiers. Where the anchor type is [OPENTIMESTAMPS], the upgrade from the initial calendar attestation to the Bitcoin block attestation MUST be completed within the 7-day profile-imposed bound, and the upgraded proof MUST be retained for the applicable retention window per the second paragraph of this section.

Each entry in the top-level anchors array is an object with the following members.

type: REQUIRED string discriminator. MUST be one of rfc3161 or opentimestamps.

value: REQUIRED on every anchor entry. The anchor token bytes, base64-encoded. For rfc3161 the value is the base64 encoding of the full TimeStampResp DER bytes (sufficient for offline cryptographic re-verification by a holder with access to the TSA's published public key). For opentimestamps the value is the base64 encoding of the OpenTimestamps proof blob (the .ots serialization). A verifier MUST cryptographically re-verify the anchor against the signed envelope using these bytes per Section 9.1; anchor entries served without value MUST NOT be reported as anchor_valid_*=true.

status: OPTIONAL informational string. When present, MUST be one of anchored (the anchor has reached its final attestation state: an [RFC3161] token has been obtained, or an [OPENTIMESTAMPS] commitment has upgraded to its Bitcoin block attestation), pending (the anchor has been requested but the final attestation state has not yet been reached, e.g. an OpenTimestamps commitment that has been submitted to a calendar but has not yet upgraded to a Bitcoin block within the 7-day bound of this section), or failed (the anchor submission was attempted and did not produce a usable attestation, e.g. a TSA returned an error response or an OpenTimestamps calendar refused the commitment). status is operational metadata; a verifier MUST NOT derive cryptographic validity from status alone, and MUST always re-verify the value bytes per Section 9.1.

bitcoin_block: OPTIONAL informational string. The Bitcoin block hash at which an [OPENTIMESTAMPS] commitment was anchored. Present only on entries with type=opentimestamps and status=anchored; absent on rfc3161 entries and on pending or failed OpenTimestamps entries. bitcoin_block is operational metadata; a verifier MUST NOT derive cryptographic validity from bitcoin_block alone, and MUST always re-verify the value bytes against the OpenTimestamps proof per Section 9.1.

An OPTIONAL envelope-level witness_policy member, a sibling of payload, signature, and anchors, declares an N-of-M durable-anchoring quorum over the anchors array. witness_policy is an object with two members: required, a REQUIRED integer in the closed range [1, length of witnesses]; and witnesses, a REQUIRED non-empty JSON array whose values are a subset of the anchor type vocabulary {rfc3161, opentimestamps}. The witnesses array MUST NOT contain duplicate values and MUST NOT contain any value outside that vocabulary; in particular a transparency-log pointer such as Rekor is NOT a witness type and MUST be rejected if it appears in witnesses. The policy

declares that the receipt's durable anchoring is satisfied only when at least required distinct witness types named in witnesses each hold a verifiable inclusion proof, that is, an anchor entry of that type whose value bytes re-verify against SHA-256(JCS(envelope_minus_anchors)) per Section 9.1 and, for opentimestamps, has upgraded to its Bitcoin block attestation within the 7-day bound of this section.

A receipt reaches the quorum-met state (the reference implementation reports this as witness_quorum_met) only when the count of distinct witness types satisfying the preceding paragraph is greater than or equal to required. A Compliance Receipt MUST NOT assert that durable anchoring has been achieved, and a producer MUST NOT set or report witness_quorum_met, unless required witnesses each hold a real, verifiable inclusion proof; an anchors entry with status=pending or status=failed, or with absent or non-verifying value bytes, does NOT count toward the quorum. This is the same false-attestation principle that governs the rest of this profile: a receipt MUST NOT claim a cryptographic property it cannot prove from retained bytes, and a verifier MUST recompute the quorum from the re-verified anchors entries rather than trust any producer-asserted quorum flag. witness_policy places no constraint on a receipt that carries no such member; the baseline single-anchor requirement of this section continues to apply to every Compliance Receipt regardless of whether witness_policy is present.

5.5. Extension Fields

This profile registers extension fields across five groupings that MAY appear in the signed payload object alongside the fields defined by [ACTA-RECEIPTS]: (a) regulatory classification fields (risk_class, incident_class) defined in this section; (b) the cross-agent envelope-binding field counterparty_binding defined in Section 5.6; (c) per-action freshness and integrity fields (result_digest, expires_at, nonce, tool_fingerprint, config_manifest_digest, cve_inventory_digest) and build-provenance fields (executable_hash, sbom_digest, slsa_provenance_pointer, supply_chain_pointer) defined in Section 5.7 and Section 5.8; (d) server-built enforcement-attestation fields (authorized_under_mandate, controls_evaluated) defined in Section 5.9; and (e) self-declared threat-framework taxonomy fields (mitre_techniques, mitre_atlas, owasp_llm_top10, nist_ai_rmf, iso_42001, eu_ai_act_articles), the opaque caller-supplied rfc3161_timestamp token, and the platform-set guard framework_mappings_self_declared defined in Section 5.10. All extension fields appear inside the signed payload object and are therefore covered by the upstream Section 5.6 signature scope.

risk_class: A vocabulary term identifying the risk classification of

the Action under the Deployer's risk management documentation. The vocabulary MUST be referenced in the Audit Pack metadata. Where the Deployer operates under [EU-AI-ACT], the documentation is the Provider's Article 9 risk management system as referenced via the instructions for use under Article 26(1); where the Deployer operates under [COLORADO-AI-ACT], the documentation is the Section 6-1-1703(2) risk management policy and program.

incident_class: A vocabulary term identifying the incident classification of the Action under the applicable regime: an ICT-related incident under [DORA], with classification criteria in [REG-2024-1772] and the canonical reporting enumeration of Annex II data glossary, field 3.23 (Type of the incident) of [REG-2025-302] (verifiers MUST resolve the canonical values from the regulation directly); a Cybersecurity Event under 23 NYCRR 500.1(f) (or, where the Section 500.17(a) reporting threshold is met, a Cybersecurity Incident under 23 NYCRR 500.1(g)) for Covered Entities of [NYDFS-500]; a Covered Cyber Incident under [CIRCIA] once the final rule takes effect; or a security incident under 45 CFR 164.304 for Covered Entities of [HIPAA-SECURITY]. Implementations MAY refine the set, provided the flattened mapping in the Audit Pack manifest (Section 8) projects each refinement to the applicable canonical category for each in-scope regime.

risk_class MUST be encoded as a JSON string. incident_class MUST be encoded as a JSON string drawn from the canonical vocabulary referenced in the Audit Pack, OR as a JSON array of such strings to preserve cross-regime classification (for example, a single Action that is both a DORA ICT-related incident and a CIRCIA Covered Cyber Incident, or both a NYDFS Cybersecurity Incident and a CIRCIA Covered Cyber Incident). Both fields are OPTIONAL at the syntactic level but MAY be REQUIRED by the regime bindings in Sections 5 and 6 of this document.

Implementations MAY define additional extension fields. Such fields MUST NOT collide with names defined by [ACTA-RECEIPTS] or by this document. Implementations defining extension fields SHOULD register them in the registry described in Section 11.

5.6. Counterparty Binding

This section is normative. counterparty_binding is an in-payload object an acknowledging agent ("B") emits to carry a cryptographic digest of the full signed envelope of an originating agent ("A"). It provides cross-agent byte-equality evidence when a shared intermediary sits between two honest agents and the per-agent hash chains of Section 5.3 validate independently regardless of whether B's observed bytes equal A's signed bytes. action_ref is a

correlation anchor, not a cryptographic binding ([ACTA-RECEIPTS] Section 2.2); counterparty_binding moves the evidence onto B's own COSE or JWS signature, which the verifier already trusts.

5.6.1. Wire Shape

The field MUST appear inside the signed payload object. It MUST NOT appear in unprotected COSE or JWS header parameters, or in external_aad per [RFC9052] Section 4.3 when the receipt is used for audit (external_aad is permissible only in transport-optimized modes out of scope for Compliance Receipts). For COSE-framed receipts the field sits inside the COSE_Sign1 or COSE_Sign payload per [RFC9052] Section 4.1; for JWS-framed receipts it is a top-level claim per [RFC7515].

The field is an object with the following members.

envelope_hash: REQUIRED string. Base64-encoded SHA-256 digest computed over A's entire serialized signed envelope, including A's signature bytes. The digest input is framing-specific:

* JSON-framed (this profile's default for receipts not transported under COSE or JWS, and mandatory-to-implement for any conformant Compliance Receipt implementation): the JCS-canonical UTF-8 byte sequence of A's signed envelope JSON object per [RFC8785], where the envelope is the three-key object {"payload": <signed payload object>, "signature": <signature object with alg, kid, sig members>, "anchors": <array of anchor objects, OPTIONAL>}. The payload object carries the signed fields A emitted (including type, issuer_id, issued_at, action_ref, payload_digest, previousReceiptHash, decision, and any extension fields under Section 5.5); the signature object carries the algorithm identifier, key identifier, and base64- or base64url-encoded signature bytes exactly as A emitted them. B MUST NOT re-canonicalize A's payload or strip the anchors array before computing the digest.

* COSE-framed: the full COSE_Sign1 or COSE_Sign byte string per [RFC8949] Section 4.2 (deterministic encoding).

* JWS-framed: the full JWS Compact Serialization (header.payload.signature) after payload canonicalization per [RFC8785].

The digest algorithm is SHA-256 (mandatory-to-implement). The encoding MUST be standard base64 per [RFC4648] Section 4 on emission, OR base64url per Section 5 where the surrounding transport requires URL-safe encoding; verifiers MUST accept both

alphabets and MUST normalise to a single alphabet (typically standard base64) before byte-comparing to a recomputed value. Including A's signature in the digest scope binds the signed-over content of A's receipt at the envelope level and prevents an intermediary that re-signs A's claims with a different key from escaping detection.

The framing in which A's envelope was emitted MUST be preserved through B's binding. The value of envelope_hash is framing-specific because JCS-canonical JSON (UTF-16 code-unit lexicographic key ordering per [RFC8785]), COSE deterministic encoding (length-then-byte map-key ordering per [RFC8949] Section 4.2), and JWS Compact Serialization with JCS-canonical payload (UTF-16 code-unit lexicographic ordering per [RFC8785]) produce different byte sequences from the same semantic payload-and-signature, and the three framings therefore yield different envelope_hash values for the same underlying receipt. A transcoding intermediary that re-frames A's envelope (JSON to COSE, COSE to JWS, or any other pairing) changes the digest input and MUST be treated as a tampering event by the verifier; verifiers MUST NOT reframe an envelope before recomputing envelope_hash.

Future revisions MAY extend to additional digest algorithms drawn from the [ACTA-RECEIPTS] digest algorithm registry; implementations that require algorithm negotiation SHOULD carry the algorithm identifier out of band in the Audit Pack manifest rather than in the wire field.

receipt_ref: REQUIRED opaque content-addressed locator the verifier resolves through the Audit Pack or a Deployer-published index to A's full signed envelope. The value is an opaque string from the verifier's perspective; producers MAY use any stable identifier scheme (URI, content-addressed digest, opaque database id) so long as the Audit Pack resolution layer returns the correct envelope bytes. Future profiles (for example, a SCITT-style inclusion-proof profile under [ACTA-RECEIPTS] Section 4.2 extension semantics) MAY layer on this field.

expect_ack_from: OPTIONAL string. The expected acknowledging-party

identifier, expressed as a kid or issuer_id value matching the same bare-identifier form required by Section 5.1.3. When present, the field declares which acknowledging party A or the producer expects to sign over this receipt's bytes; a verifier MAY use expect_ack_from to cross-check that the acknowledging receipt's kid matches the expected identifier. Verifiers MUST NOT reject solely on absence of an acknowledging receipt; absence is a liveness-loss signal observable through Audit Pack metadata rather than a non-conformance condition on the current receipt.

transport_label: OPTIONAL string (mcp, bus, orchestrator, http). Operational only; verifiers MUST NOT derive trust from this label.

```
"counterparty_binding": {
  "envelope_hash": "bDqg...5PE=",
  "receipt_ref": "asqav-receipt://org/123/agent_A/seq/4811",
  "expect_ack_from": "00000000000000000098",
  "transport_label": "mcp"
}
```

The COSE form follows the same member set under deterministic CBOR map ordering per [RFC8949] Section 4.2.

5.6.2. Emitter Behaviour

B SHOULD emit counterparty_binding when any of the following hold: A's signing request flagged the action as requiring acknowledgment (for example, by populating an expect_ack_from list); the Deployer's risk management documentation requires bilateral byte-binding; or B is operating under the guidance of Section 10.11. B MUST compute envelope_hash over the exact byte stream it received and accepted, not over a re-canonicalization at B; re-canonicalizing at B masks intermediary tampering whenever the tampered bytes canonicalize to the same payload object, which is the threat case this section addresses. Where one acknowledgment receipt confirms envelopes from N originators, the field MAY be an array of objects; pairwise bindings cannot prove all N originators emitted identical bytes (see Section 10.10).

5.6.3. Verifier Behaviour

A Compliance Verifier processing a receipt carrying counterparty_binding MUST, in addition to Section 9.1, resolve receipt_ref through the Audit Pack or a Deployer-published index to A's full signed envelope, recompute the SHA-256 digest of that envelope under the scope rule of Section 5.6.1, base64-encode the result, and compare to envelope_hash. A non-resolving receipt_ref or a digest mismatch MUST cause the acknowledging receipt to be reported

non-conformant; liveness loss at A MUST NOT be silently treated as success. Where expect_ack_from is present, the verifier SHOULD additionally check that the acknowledging receipt's signature.kid matches the declared identifier (under the bare-identifier form required by Section 5.1.3); mismatch SHOULD be reported as an axis flag rather than as outright non-conformance because the field is OPTIONAL.

The Deployer or Audit Pack producer MUST retain A's signed envelope for at least as long as any acknowledging receipt binding it remains within retention under Sections 5 and 6. For chains of three or more agents, this profile defaults to pairwise bindings; multi-signer co-presence under [RFC9052] Section 4.1 is OPTIONAL, and verifiers MUST NOT treat a co-signed envelope as a substitute for a pairwise binding chain.

5.7. Result-Bound and Freshness Extensions

This section is normative. It defines six OPTIONAL extension fields that may appear inside the signed payload object to bind the receipt to the byte-equality of a downstream result, to bound the freshness of a decision, to declare the tool and configuration that produced the action, and to record the supply-chain Common Vulnerabilities and Exposures (CVE) inventory in effect at signing time. The fields are independently OPTIONAL; an implementation MAY emit any subset. All six are covered by the upstream Section 5.6 signature scope under [ACTA-RECEIPTS].

result_digest: OPTIONAL object of the same shape as payload_digest defined in Section 2.2 of [ACTA-RECEIPTS]: REQUIRED string hash formatted sha256:<64 lowercase hex chars>, REQUIRED integer size in bytes, OPTIONAL string preview. The digest covers the canonicalized bytes of the downstream Action's result body (response payload, tool output, model completion). The field is emitted on a follow-up protectmcp:observation:result_bound receipt that references the originating protectmcp:decision via action_ref; a verifier processing a result-bound observation MUST treat a digest mismatch between result_digest and the verifier's local recomputation over retained result bytes as a non-conformance condition. Result bytes covered by result_digest are subject to the same retention floor as the originating decision receipt under Sections 5 and 6.

expires_at: OPTIONAL ISO 8601 timestamp with explicit timezone, encoded as a JSON string. Declares the wall-clock time after which the producing system considers the decision result stale and not safe to replay. The field provides an upper freshness bound that is additive to the 300-second forward-skew bound on issued_at

of Section 5.1.2; expires_at bounds replay safety from above, the forward-skew rule bounds emission honesty from above. A verifier MUST reject a downstream action that replays a decision whose expires_at lies in the past relative to the replay's wall clock; verifiers MUST NOT reject the originating receipt itself solely because expires_at has elapsed (the receipt remains valid as a record of the decision at issued_at).

nonce: OPTIONAL JSON string carrying a producer-generated value that is unique across the producer's emission stream for the lifetime of the cryptographic key identified by kid. The field SHOULD be a base64url-encoded random value of at least 16 bytes, OR a structured identifier (UUIDv4, UUIDv7, ULID) the producer guarantees globally unique. Verifiers SHOULD reject a second receipt that carries the same nonce under the same issuer_id as a replay candidate; the rejection is informational where the bound action is idempotent and load-bearing where the bound action is not. The field is OPTIONAL at the syntactic level but is a SHOULD-emit for any producer whose downstream actions are not idempotent.

tool_fingerprint: OPTIONAL JSON string formatted sha256:<64 lowercase hex chars> over the canonical declaration of the tool that produced the Action, including the tool's name, version, declared input schema, declared output schema, and any tool-specific configuration the producer treats as part of the tool's identity. The canonicalization rule for the declaration MUST be JCS per [RFC8785]. The field binds a receipt to a specific tool identity; a verifier or auditor reproducing the Action can detect tool drift (the same tool name with a different version, schema, or configuration) by comparing fingerprints across receipts in the same chain. The field is OPTIONAL and complementary to action_ref: action_ref identifies the call, tool_fingerprint identifies the callee.

config_manifest_digest: OPTIONAL JSON string formatted sha256:<64 lowercase hex chars> over the canonical bytes of the producer's configuration manifest in effect at the time the Action was signed. The manifest content is operator-defined and SHOULD include the producer's policy bundle reference, model identifiers and versions, prompt template digests, retrieval index identifiers, and any other inputs whose change would constitute a substantial modification of the producing system under Article 43 of [EU-AI-ACT] or under Section 6-1-1701 of [COLORADO-AI-ACT]. The field is OPTIONAL but, when emitted, SHOULD resolve through the Audit Pack to retained manifest bytes for the duration of the longest applicable retention floor in Sections 5 and 6.

cve_inventory_digest: OPTIONAL JSON string formatted sha256:<64 lowercase hex chars> over the canonical bytes of the producer's CVE inventory at the time the Action was signed. The inventory content SHOULD list the CVE identifiers known to apply to the producer's executing image and its declared runtime dependencies, plus the producer's accepted-residual rationale per [EU-AI-ACT] Article 15 robustness obligations or the equivalent obligations under Sections 5 and 6. The field binds a snapshot of the producer's known-vulnerability surface to the receipt; a regulator examining the receipt can resolve the digest through the Audit Pack to the canonical inventory bytes that were in effect when the Action was signed, rather than relying on a later-time inventory that may have been updated after the Action was performed.

Implementations emitting result_digest SHOULD use the dedicated protectmcp:observation:result_bound type registered in Section 11.2 for the follow-up receipt that carries the bound digest. Implementations MAY emit expires_at, nonce, tool_fingerprint, config_manifest_digest, and cve_inventory_digest on any receipt type defined by this profile; the fields are type-agnostic.

5.8. Build-Provenance Extensions

This section is normative. It defines four OPTIONAL extension fields that bind the receipt to the supply-chain provenance of the executable that produced the Action. The four fields form a layered subsumption set: executable_hash binds the running binary, sbom_digest binds the dependency manifest the binary was built from, slsa_provenance_pointer resolves to the SLSA attestation envelope for that build, and supply_chain_pointer resolves to a transparency-log entry (in-toto, Sigstore, or Rekor) covering the build. An implementation MAY emit any subset. All four are covered by the upstream Section 5.6 signature scope under [ACTA-RECEIPTS].

executable_hash: OPTIONAL JSON string formatted sha256:<64 lowercase hex chars> over the canonical bytes of the executable that invoked the Action. For container-based producers the canonical bytes are the immutable image manifest digest of the running image (the value an OCI registry returns under the same name plus tag, computed under the OCI Image Manifest Specification). For non-container executables the canonical bytes are the SHA-256 of the on-disk binary file at the path the producer's runtime resolved. The field binds build-side provenance into the signed receipt: a regulator examining the receipt can recover the exact executable identity that produced the Action without trusting any side-channel attestation. A verifier MAY cross-check executable_hash against the executable identity declared in the SLSA attestation resolved through slsa_provenance_pointer; mismatch SHOULD be

reported as an axis flag rather than as non-conformance because the two fields may identify the same artefact under different addressing schemes.

sbom_digest: OPTIONAL JSON string formatted sha256:<64 lowercase hex chars> over the canonical bytes of the Software Bill of Materials (SBOM) document covering the executing image. The canonical form MUST be either CycloneDX (any 1.x specification version, JSON form, with the canonicalization rule defined by CycloneDX itself) or SPDX (version 2.x or later, JSON form, with the canonicalization rule defined by SPDX itself); the producer SHOULD record the chosen format and version in the Audit Pack manifest entry for the receipt so a verifier can recompute the digest. The field complements executable_hash: executable_hash identifies the artefact, sbom_digest identifies the dependency closure of that artefact.

slsa_provenance_pointer: OPTIONAL JSON string carrying an https URL that resolves to the Supply-chain Levels for Software Artifacts (SLSA) provenance attestation envelope for the build that produced the executable identified by executable_hash. The pointer target SHOULD be the SLSA Provenance v1.0 in-toto statement form; producers MAY emit earlier SLSA versions where toolchain support is incomplete, and verifiers SHOULD accept any SLSA version they implement. The field shifts the trust root for build-side provenance off the producer’s self-attestation and onto the build platform’s attestation; the verifier’s confidence in the resolved attestation is bounded by the verifier’s trust in the build platform’s signing root.

supply_chain_pointer: OPTIONAL JSON string carrying an https URL that resolves to a transparency-log entry covering the build that produced the executable identified by executable_hash. The pointer target SHOULD be an in-toto attestation, a Sigstore entry, or a Rekor entry, in that preference order where the producer can choose; verifiers SHOULD accept any of the three. The field provides a transparency-log-backed audit path for the build, independent of the SLSA attestation envelope referenced by slsa_provenance_pointer; the two fields are complementary because a transparency-log entry attests inclusion under a public log, while a SLSA attestation attests build-platform output bytes.

The four fields are type-agnostic and MAY appear on any receipt type defined by this profile. Where a Deployer operates under a regulatory regime that requires build-side traceability (Article 12 of [EU-AI-ACT] read together with Article 17 of [DORA]; the audit-controls obligation of 45 CFR 164.312(b) of [HIPAA-SECURITY]; the recordkeeping rule of [SEC-17A-4] read with [NYDFS-500] 23 NYCRR

500.6), the Deployer SHOULD emit at least executable_hash on every protectmcp:decision receipt and SHOULD retain the SBOM, the SLSA attestation, and the transparency-log entry through the longest applicable retention floor in Sections 5 and 6.

5.9. Enforcement-Attestation Extensions

This section is normative. It defines two OPTIONAL extension fields that record, inside the signed payload object, which authorization and enforcement controls the issuing platform genuinely evaluated when it signed the receipt. Both fields are server-built: they are populated by the issuing platform at signing time, never carried in the producer's signing request, and a caller-supplied value for either field MUST be dropped by the issuing platform before signing. Both fields are covered by the upstream Section 5.6 signature scope under [ACTA-RECEIPTS]. The design rule for both fields is omission-over-false attestation: a control that did not run is represented by the absence of its key, never by a present key asserting a result the control did not produce. The two fields each carry a false-attestation guard that rejects a present-but-malformed attestation, in the same spirit as the framework_mappings_self_declared guard of Section 5.5 and the witness_policy quorum guard of Section 5.4.

authorized_under_mandate: OPTIONAL object recording that the Action was signed under a self-declared authorizing mandate. The object carries four members: mandate_id (REQUIRED string, the issuer-scoped identifier of the mandate the Action was authorized under), issuer_id (REQUIRED string, the identifier of the party that issued the mandate, in the bare-identifier form required by Section 5.1.3), scope_digest (REQUIRED string formatted sha256:<64 lowercase hex chars> over the canonical bytes of the mandate's authorized-action-types scope), and verified (REQUIRED boolean). The trust semantics are deliberately narrow: verified=true asserts self-declared issuer authority, the same trust level as framework_mappings_self_declared of Section 5.5, and is NEVER an issuing-platform attestation of verified third-party authorization. The mandate binding is self-declared by the issuer, is evaluated against the issuing platform's own clock at signing time, and scopes the Action to a set of authorized action types; this profile does NOT define a value cap, a counterparty restriction, or any other constraint on the mandate, and a verifier MUST NOT infer one from the presence of this field. A verifier resolves scope_digest by retrieving the mandate identified by mandate_id through the Audit Pack or a Deployer-published mandate index and recomputing the digest over the canonical scope bytes; a mismatch MUST be reported as a non-conformance condition. The false-attestation guard for this field rejects an authorized_under_mandate object that is present but

does not carry all of mandate_id, issuer_id, verified=true, and a well-formed scope_digest: a present-but-malformed attestation is rejected at signing time rather than signed and surfaced as truth.

controls_evaluated: OPTIONAL object enumerating the enforcement controls that genuinely fired when the issuing platform signed the Action, plus the allow result. The member keys are drawn from a closed set: emergency_halt, delegation_scope, quorum, mandate, policy, content_scan, and result; an unknown key MUST be rejected. Each control key is present ONLY when its control actually ran on this sign; an absent key means the control never ran on this sign, and a verifier MUST NOT infer from an absent key that the control ran and passed silently. The quorum member, when present, MUST carry fired=true together with a 64-lowercase-hex attestation_hash proving the quorum evaluation; the policy member, when present and asserting a policy was evaluated, MUST carry matched_count greater than or equal to 1. The false-attestation guard for this field rejects a present-but-malformed controls_evaluated object: an unknown control key, a quorum member lacking fired=true plus a 64-hex attestation_hash, or a policy member asserting evaluation without matched_count greater than or equal to 1, is rejected at signing time. Because the field is server-built and a caller-supplied controls_evaluated is dropped before signing, a verifier MAY treat the enumerated keys as the issuing platform’s own record of which controls it ran.

Both fields are type-agnostic and MAY appear on any receipt type defined by this profile, though they are most commonly emitted on protectmcp:decision receipts where an authorization or enforcement evaluation produced the recorded decision. Neither field replaces the policy-evaluation honesty rule that an issuing platform MUST NOT assert a control ran when it did not (the design note carried under Section 10): controls_evaluated records which controls ran, not that any control blocked, and an absent control key is the conformant representation of a control that did not run.

5.10. Threat-Framework Taxonomy Extensions

This section is normative. It defines six OPTIONAL caller-supplied taxonomy fields, one OPTIONAL caller-supplied opaque timestamp token, and one platform-set false-attestation guard boolean, all of which MAY appear inside the signed payload object. The six taxonomy fields record producer-asserted mappings of the Action into established threat-and-control catalogues; they are self-declared and are NOT verified by the issuing platform. The guard boolean exists so that a verifier can tell a self-declared classification apart from a platform-verified one. All eight fields are covered by the upstream Section 5.6 signature scope under [ACTA-RECEIPTS], and an

implementation MAY emit any subset.

mitre_techniques: OPTIONAL JSON array of MITRE ATT&CK technique identifiers (for example T1059, T1078) self-declared by the producer. The values are referenced by identifier from the MITRE ATT&CK enterprise matrix. The field is not verifier-checked by the issuing platform; when the array is populated the issuing platform MUST set framework_mappings_self_declared to true.

mitre_atlas: OPTIONAL JSON array of MITRE ATLAS identifiers (for example AML.T0051) covering AI-system-specific adversary techniques, self-declared by the producer and referenced by identifier from the MITRE ATLAS catalogue. When populated the issuing platform MUST set framework_mappings_self_declared to true.

owasp_llm_top10: OPTIONAL JSON array of OWASP Top 10 for LLM Applications identifiers (for example LLM01, LLM02), self-declared by the producer and referenced by identifier from the OWASP Top 10 for LLM Applications publication. When populated the issuing platform MUST set framework_mappings_self_declared to true.

nist_ai_rmf: OPTIONAL JSON array of NIST AI Risk Management Framework function identifiers and subcategories (for example GOVERN-1.1, MEASURE-2.7), self-declared by the producer and referenced from [NIST-AI-RMF]. When populated the issuing platform MUST set framework_mappings_self_declared to true.

iso_42001: OPTIONAL JSON array of ISO/IEC 42001:2023 control identifiers (for example A.6.2.6), self-declared by the producer and referenced from ISO/IEC 42001:2023. When populated the issuing platform MUST set framework_mappings_self_declared to true.

eu_ai_act_articles: OPTIONAL JSON array of EU AI Act article identifiers (for example Article-12, Article-15), self-declared by the producer and referenced from [EU-AI-ACT]. When populated the issuing platform MUST set framework_mappings_self_declared to true.

rfc3161_timestamp: OPTIONAL JSON string carrying a base64-encoded [RFC3161] TimeStampResp (DER) supplied by the producer at signing time and preserved verbatim on the receipt for offline TSA chain verification independent of any platform-issued anchors. The payload entry is an opaque caller-supplied token, NOT the per-receipt anchor produced by the platform under Section 5.4; the base64 encoding is per [RFC4648]. This field does not flip framework_mappings_self_declared.

framework_mappings_self_declared: OPTIONAL JSON boolean false-attestation guard set by the issuing platform. The platform MUST set it to true whenever any of mitre_techniques, mitre_atlas, owasp_llm_top10, nist_ai_rmf, iso_42001, or eu_ai_act_articles is populated. A producer-supplied value of false alongside a populated taxonomy field MUST be overridden by the issuing platform, in the same spirit as the false-attestation guards of Section 5.9 and the witness_policy quorum guard of Section 5.4. The guard does not assert that the self-declared mappings are correct; it asserts only that they are self-declared rather than platform-verified, and a verifier MUST NOT treat a populated taxonomy field as platform-verified.

The eight fields are type-agnostic and MAY appear on any receipt type defined by this profile.

6. European Union Bindings

6.1. EU AI Act Article 12 Binding

Each subsection cites the operative phrase of Article 12 and binds it to the receipt field that satisfies it.

6.1.1. Article 12(1), automatic recording of events

Article 12(1) requires High-Risk AI Systems to technically allow for the automatic recording of events (logs) over the lifetime of the system. The signed-receipt format provides one mechanism that satisfies that logging capability; alternative mechanisms remain valid. Where this profile is chosen, a Compliance Receipt SHOULD be produced for every Action against an external resource, and a configuration change that disables receipt generation SHOULD be recorded as a protectmcp:lifecycle Compliance Receipt. Implementations MAY emit at finer or coarser granularity so long as the log set, taken together, satisfies Article 12(2)(a) through (c).

6.1.2. Article 12(2)(a), identifying situations that may result in the high-risk AI system presenting a risk within the meaning of Article 79(1) or in a substantial modification

The combination of type, decision, reason, and policy_digest MUST be sufficient for an auditor to identify, by query alone, receipts that correspond to risk situations enumerated in the Deployer’s risk management documentation. Where the Deployer classifies an Action as risk-bearing, the receipt MUST carry a risk_class extension field.

6.1.3. Article 12(2)(b), facilitating the post-market monitoring referred to in Article 72

The hash-chain linkage required by Section 5.3 satisfies post-market monitoring traceability. The chain head MUST be made available to the Provider and to the competent authority on request.

6.1.4. Article 12(2)(c), monitoring the operation of high-risk AI systems referred to in Article 26(5)

Any change to the policy artefact referenced by policy_digest MUST produce a new digest. A change in policy_digest between two otherwise-comparable Actions may be examined by the Deployer or by a regulator as a candidate substantial-modification event under Article 43, and MUST be retained at least as long as the longest receipt in the chain that references either digest.

6.1.5. Retention

Article 12 itself sets no retention period; the operative deployer floor is Article 26(6) ("at least six months"). The parallel provider floor in Article 19(1) sets the same six-month minimum on Providers; this profile’s retention bindings are written from the Deployer perspective, and a Provider that wishes to use Compliance Receipts as its Article 19(1) record SHOULD adopt the Deployer floor explicitly through a separate Provider-role binding (deferred to a future revision).

EU AI Act Article 26(6) requires six months of logs (interpreted as 184 days when expressed as a day-count floor for hash-chain anchoring intervals). The normative requirement on Compliance Receipts is: implementations MUST retain receipts until the later of (a) the day six calendar months after the date of the Action, computed calendar-arithmetically per [ISO8601-2] duration arithmetic, and (b) any longer Union or national law floor. The six-month period is read calendar-month-wise (the receipt expiry day is the same day-of-month six months later, with end-of-month rollover where the target month is shorter), not as a fixed day count.

The 184-day figure (the maximum number of days in any rolling six-calendar-month window, worst case Aug-Jan, 31+30+31+30+31+31) is informative only; it expresses a safe day-count floor for hash-chain anchoring intervals and Audit Pack export windows where calendar-arithmetic is impractical at the producer layer. A Deployer that retains receipts strictly under the calendar-month rule above satisfies Article 26(6); a Deployer that uses 184 days as an internal day-count overestimate also satisfies it. A 183-day day-count floor is not endorsed by this profile: under a rolling six-calendar-month

window 1 August to 31 January spans 184 days and 183 days is one day short. Where the Deployer is also a Financial Entity, the sectoral floor in Section 6.3.4 applies.

6.2. EU AI Act Article 26 Binding

6.2.1. Article 26(1), in accordance with the instructions for use

policy_digest MUST resolve through Section 8 to a retained artefact (machine check). The Deployer SHOULD demonstrate consistency with the Provider's instructions for use (process check). Inability to perform the machine check is presumed non-compliance.

6.2.2. Article 26(2), assign human oversight

For any Action whose decision is allow and which the Deployer's risk management documentation marks as requiring human oversight, the Deployer MUST ensure that the receipt is either reviewed by a designated natural person within the period required by national law, or that a follow-on protectmcp:lifecycle Compliance Receipt records the absence of such review with a reason code. Both records MUST themselves be Compliance Receipts. This profile addresses the trigger and record of oversight; the competence, training, authority, and necessary support of the reviewer required by Article 26(2) remain the Deployer's separate responsibility.

6.2.3. Article 26(5), monitor the operation

A Deployer MUST be able to produce an Audit Pack covering any contiguous time window since the High-Risk AI System became operational.

6.2.4. Article 26(6), keep the logs for at least six months

Compliance Receipts under this binding MUST be retained for at least the period stated in Section 6.1.5. Where the Deployer is also a Financial Entity, the longer sectoral floor in Section 6.3.4 applies.

6.3. DORA Article 17 Binding

6.3.1. Article 17(1), ICT-related incident management process

A Compliance Receipt produced inside a Financial Entity's ICT environment may serve as the canonical record of an Action that triggered an ICT-related incident. action_ref MUST be carried into the Financial Entity's incident workflow as the primary correlation key.

6.3.2. Article 17(2), record all ICT-related incidents and significant cyber threats

The hash chain required by Section 5.3 supports the recording obligation of Article 17(2) by making after-the-fact alteration of recorded incidents detectable. The Financial Entity MUST be able to produce, on request, the chain segment covering the period of an incident, together with the anchor evidence that fixes the chain to wall-clock time.

6.3.3. Article 17(3)(b), establish procedures to identify, track, log, categorise and classify ICT-related incidents

For Actions identified as part of an ICT-related incident, the producing system MUST emit incident_class. The classification criteria are those set out in Article 18(1) of [DORA], with further specification in [REG-2024-1772]. The canonical reporting enumeration to which incident_class flattens is bound by Annex II field 3.23 of [REG-2025-302] (see Section 5.5). Implementations MUST publish a flattened mapping in the Audit Pack manifest as required by Section 5.5.

6.3.4. Retention

Article 17 of [DORA] does not itself set a uniform numeric retention floor. The five-year (1827-day) figure used by this profile derives from sectoral instruments that overlap DORA-scoped Financial Entities. Investment firms keep records of all services, activities and transactions under Article 16(6) of [MIFID2], with Article 72 and Annex I of [REG-2017-565] fixing the form and content of those records. The explicit five-year retention period in [MIFID2] is set by Article 16(7) for records of telephone conversations and electronic communications, kept for a period of five years and, where requested by the competent authority, for a period of up to seven years.

Records of customer due diligence and of transactions under Article 40 of [AMLD] are kept for five years after the end of the business relationship. The AMLD record-keeping regime is superseded, in respect of record retention, by Article 77 of [AMLR] from 10 July 2027, which preserves the five-year floor and adds a case-by-case extension up to a further five years where the competent authority so requires. Implementations operating across the AMLD-to-AMLR transition MUST satisfy whichever instrument is in force on the date of the Action.

Compliance Receipts MUST be retained for the period required by applicable Union or national law; where a sectoral floor applies, retention MUST equal or exceed the longest applicable floor. Absent a more specific rule, this profile RECOMMENDS 1827 days from the date of the Action (the worst-case rolling five-calendar-year window contains two leap days, so 1827 days satisfies "five years" regardless of the calendar years over which the window falls). Anchor evidence MUST be retained for the same period. Verification keys whose lifetime expires within the retention window MUST have their public components retained so that historical signatures remain verifiable.

7. United States Bindings

7.1. NIST AI RMF Binding

[NIST-AI-RMF] is a voluntary framework. Adoption of this profile, on its own, does not establish conformity with the AI RMF; it provides a tamper-evident receipt substrate that an AI RMF program can use as evidence under the MEASURE function and as a structured input to the GOVERN, MAP, and MANAGE functions. [NIST-GENAI-PROFILE] applies the AI RMF functions to generative AI; the profile bindings below apply to generative and non-generative AI agent deployments alike unless explicitly noted.

7.1.1. GOVERN function

The GOVERN function requires that organizations document AI policies and procedures. The combination of policy_digest and the Audit Pack manifest provides a machine-readable binding between every Action and the policy artefact in force at the time of the Action. A change to the policy artefact MUST produce a new policy_digest value (per Section 5.2.2); the Audit Pack therefore records every policy change in a tamper-evident manner.

7.1.2. MAP function

The MAP function requires that the context, capabilities, and risks of an AI system be characterised. The combination of type, tool_name, action_ref, and iteration_id SHOULD be sufficient for an auditor to reconstruct the operational context of any Action without dereferencing the underlying payload.

7.1.3. MEASURE function

The MEASURE function requires that AI risks and impacts be analysed and tracked over time. The hash-chain linkage required by Section 5.3 provides tamper-evident continuity of the receipt stream over the AI system's operational lifetime, satisfying the traceability prerequisite of MEASURE.

7.1.4. MANAGE function

The MANAGE function requires that AI risks be prioritised and acted upon based on projected impact. The risk_class extension field carries the Deployer's risk classification of the Action; together with decision, reason, and policy_digest, it supports prioritisation and incident response without requiring the verifier to re-derive risk from the underlying payload.

7.2. Colorado AI Act (SB 24-205) Binding

[COLORADO-AI-ACT] imposes deployer obligations effective June 30, 2026 (per Senate Bill 25B-004, which postponed the original February 1, 2026 effective date). The Act regulates the deployment of High-Risk AI Systems and the prevention of algorithmic discrimination.

7.2.1. Section 6-1-1703(2), risk management policy and program

Section 6-1-1703(2) requires deployers to implement a risk management policy and program for the High-Risk AI System. policy_digest MUST resolve through Section 8 to the deployer's risk management policy artefact in force at the time of the Action. Where the Deployer classifies an Action as risk-bearing under that policy, the receipt MUST carry a risk_class extension field.

7.2.2. Section 6-1-1703(3), impact assessment

Section 6-1-1703(3) requires deployers to complete an impact assessment annually and within 90 days after any intentional and substantial modification of the High-Risk AI System. The combination of type, policy_digest, and previousReceiptHash MUST be sufficient for an auditor to identify, by query alone, the receipts that span the period covered by an impact assessment, including any policy changes within that period.

7.2.3. Section 6-1-1703(7), notice of algorithmic discrimination

Where a Deployer determines that a High-Risk AI System has caused or is reasonably likely to have caused algorithmic discrimination, the producing system SHOULD record that determination as a protectmcp:lifecycle Compliance Receipt naming the determination, the affected receipts by action_ref, and the policy or risk-management response with a reason code.

7.3. Texas Responsible AI Governance Act (HB 149) Binding

[TEXAS-TRAIGA] takes effect January 1, 2026. The Act adopts an intent-based liability framework for the development and deployment of AI systems and provides a safe harbor at Section 552.105(e)(2)(D) of the Texas Business and Commerce Code for organisations that substantially comply with the most recent version of [NIST-GENAI-PROFILE], or another nationally or internationally recognized risk management framework for AI systems, and operate an internal review process.

7.3.1. Safe-harbor evidentiary support

Where a Deployer relies on the safe-harbor provision of HB 149 by substantially complying with [NIST-GENAI-PROFILE], the Audit Pack MAY be presented as evidence of that compliance. The bindings of Section 7.1 apply, with the additional Generative AI Profile bindings of [NIST-GENAI-PROFILE].

7.3.2. Prohibited-use detection

Receipts whose decision is deny with a reason code drawn from a vocabulary documenting the Act’s prohibited-use categories under Section 552.052 of the Texas Business and Commerce Code added by HB 149 (incitement or encouragement of physical self-harm including suicide, harm to another person, or engagement in criminal activity) MUST be retained for the period stated in Section 7.4.2 or the longer period required by Texas law, whichever is greater.

7.4. HIPAA Security Rule Binding (45 CFR Part 164, Subpart C)

[HIPAA-SECURITY] applies to Covered Entities (HIPAA) that handle electronic protected health information. The bindings below apply only to receipts whose underlying Actions reference electronic protected health information.

7.4.1. 45 CFR 164.312(b), audit controls

45 CFR 164.312(b) requires implementation of "hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information". The combination of type, action_ref, tool_name, and the hash-chain linkage required by Section 5.3 satisfies the recording requirement; the verification rules of Section 9.1 satisfy the examination requirement.

7.4.2. 45 CFR 164.316(b)(2), six-year retention

45 CFR 164.316(b)(2) (and in particular the subparagraph 164.316(b)(2)(i)) requires that the documentation required by 45 CFR 164.316(b)(1) be retained "for 6 years from the date of its creation or the date when it last was in effect, whichever is later". The audit-log content produced under 45 CFR 164.312(b) is not itself documentation required by 164.316(b)(1); the Security Rule does not set an explicit retention floor for individual audit-log records. By analogy with the six-year floor that 164.316(b)(2) places on the policies and procedures that govern audit-log generation, this profile applies the same six-year floor to Compliance Receipts whose underlying Actions reference electronic protected health information.

Records covered by the HIPAA Security Rule audit-trail retention MUST be retained for six years from the date of creation or the date when last in effect, whichever is later, per 45 CFR 164.316(b)(2). This profile expresses that floor as 2192 days from the later of (a) the date of the Action and (b) the date the policy artefact referenced by policy_digest ceased to be in effect: 2192 is the maximum number of days in any rolling six-calendar-year window (worst case spans two leap days, e.g. 2024-2030 contains February 29 of 2024 and 2028, yielding 6*365+2 = 2192 days). The six-year analogy floor is grounded in 164.316(b)(2); a Covered Entity that retains receipts strictly under 164.316(b)(2)(i) bound only to the policy artefact’s creation-or-cessation date MAY do so when no longer audit-log floor is established by separate Union, state, or sectoral law. Verification keys whose lifetime expires within the retention window MUST have their public components retained so that historical signatures remain verifiable.

7.5. NYDFS Cybersecurity Regulation Binding (23 NYCRR Part 500)

[NYDFS-500] applies to Covered Entities (NYDFS) operating under New York Banking, Insurance, or Financial Services Law. The bindings below apply only to receipts produced by such Covered Entities.

7.5.1. 23 NYCRR 500.6, audit trail

23 NYCRR 500.6(a) requires Covered Entities to securely maintain systems that, to the extent applicable and based on its risk assessment, (1) are designed to reconstruct material financial transactions, and (2) include audit trails designed to detect and respond to cybersecurity events that have a reasonable likelihood of materially harming any material part of the normal operations of the Covered Entity. The hash chain required by Section 5.3 together with the anchor evidence required by Section 5.4 satisfies the tamper-evidence prerequisite of the audit-trail obligation.

7.5.2. 23 NYCRR 500.17, notices to superintendent

23 NYCRR 500.17(a)(1) requires that "Each covered entity shall notify the superintendent electronically in the form set forth on the department’s website as promptly as possible but in no event later than 72 hours after determining that a cybersecurity incident has occurred at the covered entity, its affiliates, or a third-party service provider." The reporting trigger is a Cybersecurity Incident under 23 NYCRR 500.1(g), not any Cybersecurity Event under 500.1(f). For Actions identified as part of such an Incident, the producing system MUST emit incident_class with a value indicating Cybersecurity Incident under 23 NYCRR 500.1(g), and the Covered Entity MUST be able to produce, on request, the chain segment covering the period of the Incident together with the anchor evidence that fixes the chain to wall-clock time.

7.5.3. 23 NYCRR 500.6 retention

23 NYCRR 500.6(b) requires that "Each Covered Entity shall maintain records required by this section for not fewer than five years." The five-year floor applies uniformly to records required by paragraph (a)(1) (designed to reconstruct material financial transactions) and to records required by paragraph (a)(2) (audit trails designed to detect and respond to cybersecurity events that have a reasonable likelihood of materially harming any material part of the normal operations of the Covered Entity). Compliance Receipts produced under this binding MUST be retained for at least 1827 days from the date of the Action (the worst-case rolling five-calendar-year window contains two leap days).

7.6. SEC Broker-Dealer Recordkeeping Binding (17 CFR 240.17a-4)

[SEC-17A-4] applies to Broker-Dealers, Security-Based Swap Dealers, and Major Security-Based Swap Participants. The bindings below apply only to receipts produced inside such entities.

7.6.1. 17 CFR 240.17a-4(f), electronic recordkeeping system

The November 3, 2022 amendments to 17 CFR 240.17a-4 (compliance date May 3, 2023) added an audit-trail alternative to the prior write-once-read-many (WORM) electronic recordkeeping requirement. The audit-trail alternative requires that the electronic recordkeeping system permit the recreation of an original record if it is modified or deleted. The hash-chain linkage required by Section 5.3 together with the retention rule in Section 7.6.2 and the anchor evidence required by Section 5.4 satisfies the audit-trail alternative when the Compliance Receipt is the system-of-record for a regulated record.

7.6.2. 17 CFR 240.17a-4(a) and (b) retention

17 CFR 240.17a-4(a) requires preservation of certain records for not less than 6 years, the first two years in an easily accessible place. 17 CFR 240.17a-4(b) requires preservation of a different list of records for not less than three years, the first two years in an easily accessible place. Compliance Receipts that constitute or support a record listed in 17 CFR 240.17a-4(a) MUST be retained for at least 2192 days from the date of the Action, applying the same six-year worst-case methodology as Section 7.4.2; receipts that constitute or support a record listed only in 17 CFR 240.17a-4(b) MUST be retained for at least 1096 days from the date of the Action (the worst-case rolling three-calendar-year window contains one leap day). Where both apply, the longer period applies.

7.7. CIRCIA Binding (Cyber Incident Reporting for Critical Infrastructure Act of 2022)

[CIRCIA] requires Covered Entities (CIRCIA) to report Covered Cyber Incidents to the Cybersecurity and Infrastructure Security Agency within 72 hours of reasonable belief that the incident has occurred, and to report ransom payments within 24 hours. The reporting obligations take effect upon publication of the final rule. Pending publication, the bindings below apply on a voluntary basis.

7.7.1. Covered Cyber Incident reporting support

For Actions identified as part of a Covered Cyber Incident, the producing system MUST emit incident_class with a value indicating Covered Cyber Incident under [CIRCIA]. The Covered Entity MUST be able to produce, on request, the chain segment covering the period of the incident together with the anchor evidence that fixes the chain to wall-clock time.

7.7.2. Records related to a Covered Cyber Incident report

Section 2242(a)(4) of the Homeland Security Act of 2002, as enacted by [CIRCIA] and codified at 6 U.S.C. 681b(a)(4), requires Covered Entities to preserve data relevant to a Covered Cyber Incident or ransom payment in accordance with procedures established in the final rule. CISA's notice of proposed rulemaking at 89 FR 23644 (April 4, 2024), proposed Section 226.13(c), proposes a preservation period of not less than two years measured from the submission of the most recently required CIRCIA report (or the date that submission would have been required absent a preservation exception under proposed Section 226.4(a)); this profile uses that proposed floor pending publication of the final rule.

Records covered by CIRCIA preservation MUST be retained for two years from the submission of the most recently required CIRCIA report under 6 U.S.C. 681b(c)(2). Compliance Receipts that are referenced in a CIRCIA report or that the Covered Entity reasonably anticipates will be so referenced MUST be retained for the longer of (a) the period established by the final rule and (b) two years from the submission of the most recently required CIRCIA report (or the date that submission would have been required absent a preservation exception), per proposed Section 226.13(c) of the CIRCIA NPRM at 89 FR 23644 (April 4, 2024). Covered Entities MUST NOT measure the retention floor from the date of the underlying Action; an Action detected and reported months later carries a preservation window that runs forward from the report submission date.

8. Audit Pack Composition

This section is informative. It describes the contents of an Audit Pack as introduced in Section 2.

An Audit Pack contains the following items.

* The set of Compliance Receipts covered by the requested time window, in the canonical envelope form defined by [ACTA-RECEIPTS].

* The chain commitments that link the receipts: for each receipt, the value of previousReceiptHash and the recomputed digest of the predecessor envelope.

* The anchor evidence: [RFC3161] tokens, OpenTimestamps proofs, or both. Each anchor item MUST be associated, by hash, with the receipt or aggregate it covers.

* The trust anchor metadata that identifies the Deployer or other regulated entity associated with each issuer_id value.

* The verification key material for every kid value present, in a form that does not require online retrieval.

* Vocabularies referenced by reason, risk_class, incident_class, and extension fields, embedded as JSON arrays with a stable identifier. The Audit Pack MUST expose a digest-resolution facility that, given a policy_digest, returns the retained artefact.

* A regime mapping document that names which receipts the producer asserts as evidence under any of the regimes addressed by Sections 5 and 6 of this document (EU AI Act Article 12, EU AI Act Article 26, DORA Article 17, NIST AI RMF, Colorado AI Act, Texas Responsible AI Governance Act, NYDFS Part 500, HIPAA Security Rule, SEC Rule 17a-4, CIRCIA).

* The chain heads valid at the start and end of the time window, signed by the Deployer or other regulated entity.

An Audit Pack MUST itself be signed per the [ACTA-RECEIPTS] algorithm registry. The manifest MUST include bundle_digest, bundle_signature, bundle_public_key, and algorithm_registry_version.

The following manifest-level fields SHOULD appear on an Audit Pack bundle when the underlying receipt stream exposes the corresponding semantics. Each is informative and does not alter the wire shape of individual receipts.

regime_mapping_disclaimer: String emitted on bundles whose per-receipt regime predicates derive from mapping logic the original producing system did not sign. The value identifies the producer of the mapping, the document version under which it was computed, and a disclaimer that the regime-satisfaction flags are advisory and remain subject to the verifier’s own check against Sections 5 and 6.

stale_pending: Boolean flag set per bundle entry whose anchor evidence is still pending after the bound of Section 5.4 (7 days for OpenTimestamps; synchronous for RFC 3161). When true, the bundled receipt is non-conformant per Section 5.4, and a Compliance Verifier consumes the flag to drive anchor_valid_* false in its per-axis report (see Section 9.3). A verify endpoint over the same bundle SHOULD surface stale_pending in its response.

9. Verifier Behaviour

A verifier conformant to this profile is referred to as a Compliance Verifier.

9.1. Mandatory Checks

A Compliance Verifier MUST perform all of the following checks before treating a receipt as a Compliance Receipt.

* Verify the signature using the algorithm declared in signature.alg, in accordance with [ACTA-RECEIPTS].

* Resolve the verification key through one of the key-distribution mechanisms described in Section 4.3 of [ACTA-RECEIPTS] (well-known JWK Set or out-of-band distribution), or through Audit Pack trust-anchor metadata. The verifier MUST NOT trust a verification key embedded in the receipt envelope.

* Verify that all fields marked REQUIRED by Section 5 are present and well-formed.

* Verify the hash-chain linkage by recomputing SHA-256 over the canonical signing-input bytes of the immediately preceding receipt (the JCS-canonical serialization of the predecessor's signed payload object, per Section 5.3) and comparing the lowercase hex encoding to previousReceiptHash.

* Verify at least one anchor: an [RFC3161] token, an [OPENTIMESTAMPS] commitment, or both. The anchor MUST cover the signed envelope as it appears in the receipt. The verifier MUST cryptographically re-verify the anchor against the signed envelope; presence of anchor metadata without a successful cryptographic check MUST NOT yield "valid".

* Verify the future-skew bound on issued_at per Section 5.1.2. Past skew MUST NOT cause non-conformance when the receipt is within retention.

* Verify that policy_digest resolves through Section 8. A digest computed over a nonced or otherwise mixed-input form (for example, SHA-256(nonce || JCS(artefact))) MUST NOT be treated as policy_digest; the digest scope is the canonical form of the artefact alone. The verifier MUST recompute SHA-256 over the canonical form of the resolved artefact as documented in the Audit Pack manifest, and compare; for JSON artefacts the canonical form is JCS per [RFC8785].

A receipt that fails any of these checks MUST be reported as non-conformant.

9.2. Optional Checks

A Compliance Verifier MAY additionally perform any of the following.

* Cross-check the issuer_id against an external registry (LEI, EIN, CIK, NPI, GLEIF, or a Deployer-published list).

* Resolve the policy artefact referenced by policy_digest and compare it to a Provider-supplied or Deployer-supplied reference policy.

* Recompute the chain head and compare it to a Deployer-published value.

* Validate incident_class (each element if encoded as an array) and risk_class extension values against the vocabularies referenced in the Audit Pack.

9.3. Reporting

A Compliance Verifier SHOULD produce a structured per-receipt report that names the regime bindings the receipt satisfies and the outcome of the per-axis checks the verifier performed. The following fields SHOULD be emitted; the axes are independent and consumers MUST NOT collapse them into a single boolean before display.

regimes_satisfied: Array of short stable regime identifiers drawn from the regimes listed in Section 8 (for example eu_ai_act, dora, nist_ai_rmf, colorado_ai, texas_traiga, hipaa_security, nydfs_500, sec_17a4a, sec_17a4b, circia). The set is open-ended; consumers MUST treat unknown identifiers as informational. Where this field is inherited from a producer-side mapping in the bundle, the bundle-level regime_mapping_disclaimer of Section 8 applies.

anchor_valid_ots: Boolean. true when an [OPENTIMESTAMPS] anchor is present, has upgraded to the Bitcoin block attestation within the 7-day bound of Section 5.4, and re-verifies cryptographically against the signed envelope; otherwise false (including the case where the only present anchor is RFC 3161).

anchor_valid_rfc3161: Boolean. true when an [RFC3161] token is present, carries an ESSCertIDv2 per [RFC5816] where required, and re-verifies cryptographically against the signed envelope; otherwise false.

policy_digest_resolved: Boolean. true when policy_digest resolved to

a retained artefact whose JCS-canonical SHA-256 matches the receipt value, per Section 9.1; false on absence, resolution failure, or digest mismatch.

duplicate_emission_candidate: Boolean. true when at least one other receipt sharing action_ref and issuer_id has been observed in the same Audit Pack or in a verifier-maintained index; otherwise false. The axis is informational; this profile does not require verifiers to maintain a cross-receipt index. An absent index MUST report false rather than omit the axis, so consumers learn "no duplicate" from the value and learn "axis unknown" only from the verifier's documented capability set.

A receipt may carry anchor_valid_ots=true and anchor_valid_rfc3161=false (or vice versa) and still satisfy the mandatory anchor check of Section 5.4, which requires only one valid anchor. Where a receipt carries counterparty_binding per Section 5.6, the verifier SHOULD additionally emit the outcome of the check of Section 5.6.3; this profile reserves a stable field identifier for that axis pending implementation experience.

10. Security Considerations

This profile inherits all of the security considerations of [ACTA-RECEIPTS]. The following considerations are specific to the compliance binding.

10.1. Tamper Resistance

The hash-chain linkage required by Section 5.3 provides tamper-evidence at the chain level. An adversary who removes a receipt from the middle of the chain MUST recompute and re-sign every subsequent envelope. The anchor evidence required by Section 5.4 binds segments of the chain to wall-clock time, raising the cost of a re-signing attack.

Implementations SHOULD anchor at intervals no longer than 24 hours. Implementations operating under DORA Article 17, 23 NYCRR 500.17, or [CIRCIA] SHOULD anchor at intervals no longer than one hour, given the four-hour initial-notice deadline (with 72-hour intermediate-report and one-month final-report bounds) per DORA Article 17 and the RTS in [REG-2025-301], the 72-hour notification clock under 23 NYCRR 500.17, and the 72-hour CIRCIA covered-cyber-incident reporting deadline that will apply once the CIRCIA final rule takes effect.

A deployment that uses only the signature, without chain linkage and anchoring, can be rolled back by an insider with control of the signing key for the period between the deletion and the next anchor. The MUST clauses of Section 5.3 and Section 5.4 close that window.

10.2. Chain Availability Under Single-Linear Per-Agent Serialization

This section is informative. The single-linear per-agent chain requirement of Section 5.3 serializes receipt emission for a given issuer_id through a single predecessor pointer. A denial-of-service against the predecessor pointer (database row lock contention, network partition between the emitter and the predecessor store, slow IO, or an adversary deliberately holding the chain-tail lock) therefore bounds the per-agent emission throughput, because every new receipt MUST resolve the digest of the immediately prior receipt before it can be linked. A partial-write failure between predecessor-pointer commit and signature commit can additionally produce chain-head ambiguity if not handled defensively.

Issuers SHOULD use a bounded predecessor-lookup timeout (operator-tuned, typically on the order of seconds rather than tens of seconds) and SHOULD emit a structured audit event with type protectmcp:lifecycle and a stable reason code (RECOMMENDED: chain_emission_blocked) when the timeout fires, rather than silently dropping the receipt or stalling caller threads. Issuers SHOULD additionally document a chain-head recovery procedure for crashed emitters: on restart, the issuer re-reads the predecessor row, verifies that no orphan signature exists for the next sequence position, and resumes emission. Operators that require parallel per-issuer throughput beyond what a single linear chain sustains MUST use distinct issuer_id values per parallel path, with separate signing keys and chains rooted at the all-zero genesis value, per the rule in Section 5.3.

The threat profile here is availability, not confidentiality or integrity: a successful chain-availability attack delays or drops emission, but it cannot tamper with already-emitted receipts (those are protected by Section 10.1) and it cannot forge receipts (those are protected by Section 10.3). The chain_emission_blocked lifecycle receipt is itself a Compliance Receipt and therefore links into the chain once emission resumes, so the gap is detectable rather than silent.

10.3. Key Compromise

A Compliance Receipt is only as trustworthy as the key that signed it. On suspected compromise of an issuer key, the Deployer MUST publish a revocation notice that names the key, the time of suspected compromise, and the chain head at that time. Receipts signed by the compromised key after the named time MUST NOT be treated as Compliance Receipts.

Verifiers MUST consult revocation metadata supplied with the Audit Pack and MUST reject Compliance Receipts whose signing key was revoked at or before issued_at.

10.4. Retention and Long-Term Verifiability

The longest retention floor in this profile is 2192 days (six calendar years), set by Section 7.4 and Section 7.6; the EU side has a parallel five-year (1827-day) floor under Section 6.3. Both exceed the typical operational crypto-period of a signing key under recommended key-management practice. Implementations SHOULD use ML-DSA-65 from the [ACTA-RECEIPTS] algorithm registry ([FIPS204]) for receipts expected to be verified after the cryptographic lifetime of classical signature schemes ends. Implementations MUST retain public key material for the entire retention window.

10.5. Privacy

[ACTA-RECEIPTS] prohibits the inclusion of raw prompts, tool arguments, and credentials in the signed payload. This profile extends that prohibition to the extension fields defined in this document. The risk_class and incident_class values MUST be drawn from controlled vocabularies and MUST NOT carry free-text personal data.

Where the underlying Action references a data subject, the payload_digest field MUST cover the data; the data itself MUST be held in a separate store that respects the data subject’s rights under applicable law (including but not limited to the General Data Protection Regulation for EU data subjects, the California Consumer Privacy Act and Virginia Consumer Data Protection Act for the corresponding US states, and the HIPAA Privacy Rule where electronic protected health information is involved). A request for erasure that is granted under applicable data protection law MUST be reflected by deletion of the referenced payload, not by deletion of the receipt; the receipt remains as evidence that an Action occurred and was governed by a named policy at a named time.

10.6. Anchor Trust

The trust assumptions of an anchor depend on the anchor type. [RFC3161] timestamp tokens depend on the trust placed in the named Time Stamping Authority. OpenTimestamps commitments depend on the inclusion of the commitment in a public Bitcoin block. A Compliance Verifier SHOULD treat the simultaneous presence of both anchor types as stronger evidence than the presence of only one.

10.7. Replay

A Compliance Receipt is bound to a single Action via action_ref. Replay of a Compliance Receipt against a different Action is detectable by action_ref mismatch. The 300-second issued_at skew bound stated in Section 5.1.2 limits the window in which a freshly-replayed receipt can be presented as recent.

Where the verifier supports it, two receipts sharing action_ref and issuer_id SHOULD be flagged as a candidate duplicate-emission event for human review. This profile does not require verifiers to maintain a cross-receipt index; deployers needing duplicate-emission detection should arrange it at the Audit Pack production layer.

10.8. Cross-Regime Conflict

Where the same Action is in scope of more than one regime addressed by this document, the producing system MUST satisfy the union of the applicable requirements. Where a SHOULD clause in one regime conflicts with a MUST clause in another, the MUST clause prevails. Where two MUST clauses conflict, the producing system MUST refuse to issue the receipt and MUST log the refusal as a protectmcp:lifecycle Compliance Receipt.

10.9. Algorithm Agility

This profile inherits its algorithm registry from [ACTA-RECEIPTS]. Implementations MUST treat the verification of a historical receipt according to the algorithm registry that was in force at issued_at, not the registry in force at the time of verification, provided that the signing key was not revoked.

10.10. Issuer-Misrepresentation Residual

Per-agent hash chains under Section 5.3 detect tampering inside a single issuer's stream but not the cross-agent attack in which a compromised intermediary silently swaps payload bytes between two honest agents. Both per-agent chains validate; action_ref is a correlation anchor, not a cryptographic binding ([ACTA-RECEIPTS] Section 2.2). Without a cross-agent binding primitive, a regulator obtains no cryptographic answer to "did the acknowledging agent acknowledge the bytes the originating agent actually sent". This profile defines counterparty_binding (Section 5.6) as the partial mitigation; the following residuals remain.

* Endpoint collusion. If both signing keys are compromised by the same attacker, the attacker produces a coordinated forgery; no signature scheme defends against this case.

* Intermediary holds the originating agent's key. In hosted-agent deployments where the intermediary possesses the originating agent's private key, it can sign anything as either party. Remote attestation of key origin is the appropriate countermeasure and is out of scope here.

* Originator offline at verification time. Section 5.6.3 requires the originating envelope to be retrievable; if unpublished, offline, or rate-limited, the binding becomes unverifiable (liveness loss, observable as failure).

* Fan-out witness gap. When an originator broadcasts to N acknowledgers, each emits an independent pairwise binding; none witnesses any other. Append-only log profiles (future SCITT-style transparency) are deferred to a later revision.

* Key rotation orphan. If the originating agent rotates keys after emission but before an acknowledger binds it, the storage obligation of Section 5.6.3 still requires the old envelope to remain retrievable; if retention discipline fails, the binding orphans.

* Privacy of envelope hashes. envelope_hash is computed over the full signed bytes including A's signature; an observer of B's receipt learns a stable identifier for A's exact action and therefore can correlate B's behaviour across receipts even when A's payload is otherwise confidential. Where this correlation is unacceptable, a commitment scheme (for example, HMAC over the envelope with a per-counterparty key disclosed only to the verifier) is appropriate; this profile does not specify one.

* Real-time prevention. counterparty_binding is detective, not preventive: B has already accepted the bytes by the time the binding is signed. Verifiers detect tampering only at audit time; the in-flight bytes were not blocked. Where prevention is required, transport-level integrity per Section 10.11 is the appropriate primitive in addition to (not instead of) this profile.

* Payload-content semantics. The binding proves byte equality, not semantic equality. An intermediary that re-encodes A's bytes into a different but JCS-equivalent canonical form is detected (the SHA-256 differs); an intermediary that swaps A's bytes for entirely different bytes that B's policy happens to interpret as semantically equivalent is detected too. But an intermediary that swaps A's bytes for an A-signed REPLAY of a prior valid envelope from A is not detected by this binding alone; replay protection requires that the verifier also check action_ref and previousReceiptHash uniqueness within the chain segment.

10.11. Cross-Agent Integrity Trust Boundary

This section is informative. It records operator guidance for cases where channel-level protection is the only available defence and counterparty_binding per Section 5.6 has not yet been adopted by both endpoints. For channels between named principals, implementers SHOULD secure the channel using mutually authenticated TLS 1.3 per [RFC8446]; MAY use the tls-exporter channel binding per [RFC9266] derived via [RFC5705] where higher channel uniqueness is required; and MAY layer HTTP Message Signatures per [RFC9421] where intermediaries perform legitimate transformations.

Operators MUST NOT interpret transport-layer security alone as evidence of cross-agent byte equality. Only counterparty_binding produces application-layer, signed, replay-after-the-fact evidence answering that question. Topologies where the intermediary terminates TLS (CDN edges, MCP servers, message buses, orchestrators) defeat transport-layer integrity against the threat case of Section 10.10; in those topologies counterparty_binding is the only defence this profile offers, and the absence of channel-binding evidence in the Audit Pack SHOULD be documented as a known residual.

10.12. Compromised Intermediary Between Two Honest Endpoints

This section is informative. Where an Action travels from a sending agent A to a receiving agent B through one or more intermediary processes M, and where M is compromised in such a way that M presents byte sequence X to A and a different byte sequence X' to B, neither A's nor B's cryptographic signature detects the divergence in isolation: each endpoint signs the bytes it observed, and each endpoint's per-agent hash chain per Section 5.3 remains internally valid. Absent the counterparty_binding primitive this profile introduces, the only available cross-agent primitive is action_ref as a SHA-256 join key per Section 5.1.5; both A's chain and B's chain remain valid in isolation, and divergence is only recoverable through a regulator-driven post-hoc comparison of the two chains.

counterparty_binding introduced in Section 5.6 closes the case where M silently swaps bytes between two honest endpoints A and B. An acknowledging receipt under this binding is REQUIRED to carry an envelope_hash computed over the exact byte stream B received (SHA-256(A's envelope) under the digest-scope rule of Section 5.6.1). A verifier resolves receipt_ref to A's stored envelope, recomputes the digest, and compares; a mismatch indicates that the bytes B signed are not the bytes A signed, and the acknowledging receipt MUST be reported non-conformant per Section 5.6.3. The binding is detective rather than preventive: it does not stop M from performing the swap in flight, but it produces signed, replay-after-the-fact evidence that the swap occurred.

The following residuals remain and are not closed by counterparty_binding alone. The list is intentionally honest about the audit-time, not sign-time, nature of the detective evidence: a verifier resolves receipt_ref to A's retained envelope and recomputes the digest at audit time, so any residual reasoning that depends on "A is not in the loop at sign time" is rhetorical, not load-bearing.

* Collusion of M and B. If M and B are jointly compromised, M swaps the bytes in flight and B issues an acknowledging receipt carrying an envelope_hash computed over the altered bytes that B signs as if they were A's. Under counterparty_binding the audit-time verifier resolves receipt_ref to A's retained envelope and recomputes the digest, so the binding is reported non-conformant when A's storage is honest and reachable; M+B collusion alone does NOT silently succeed. M+B collusion silently succeeds only when the collusion ALSO extends to corrupting A's retained envelope, suppressing A's chain segment, or making A's storage unreachable to the auditor; that is, the true residual is M+B+(A's-storage compromise or unavailability). Operator mitigation: anchor A's chain on independent witnesses (combined [RFC3161] +

[OPENTIMESTAMPS] anchors per Section 5.4, and OPTIONAL deployer-operated transparency logs) so that A's anchored chain-segment digests are independently recoverable from public evidence; regulator-side comparison of A's anchored chain against B's stored chain detects the divergence even when A's local storage is impeached.

* Collusion of M and A. If M and A are jointly compromised, A signs a fabricated envelope at M's direction and M relays it to B; B verifies M's relay normally, B's counterparty_binding correctly digests the bytes A signed, A's per-agent chain validates, and B's per-agent chain validates. Every cryptographic invariant in this profile holds because the binding correctly attests that the bytes B received were the bytes A signed; the fraud is in A's intent, not in any byte mismatch. This residual is fundamentally outside the receipt model's threat surface: no application-layer cryptographic primitive in this profile distinguishes a fraudulent A-signed envelope from an honest A-signed envelope when M is also colluding to corroborate plausibility (relay logs, timestamping, message ordering). Operator mitigation: separation of duties between issuer (A) and intermediary (M) so that the same operator cannot control both signing keys and relay logs; anchor evidence on independent witnesses under different trust roots so that an attacker controlling A and M still cannot retroactively coordinate anchor inclusion across uncolluding timestamping authorities; out-of-band attestation by the regulator or auditor of A's operational context (provenance, code signing, runtime attestation) where the policy regime authorises it.

* Compromise of B itself. A B that has been compromised (private key extraction, supply-chain compromise, or insider operation) can sign any envelope_hash the attacker chooses; counterparty_binding proves only that the signing key acknowledged some bytes, not that those bytes match what an honest B would have observed.

* Loss of A's stored envelope. counterparty_binding requires the verifier to resolve receipt_ref to A's full signed envelope; if A's chain segment is unavailable (retention discipline failure, key rotation orphan, deliberate withholding), the binding becomes unverifiable and the receipt is reported non-conformant on liveness grounds rather than on byte-equality grounds. An adversary who can arrange A-envelope unavailability and then re-emit colluding bytes can degrade the binding from a byte-equality check to a liveness-loss flag.

Operators concerned about these residuals in the absence of single-point cryptographic defence SHOULD:

* Anchor receipts to multiple independent witnesses. Where both [RFC3161] and [OPENTIMESTAMPS] anchors are present per Section 5.4, a coordinated M-B collusion attack must also induce both timestamping authorities to anchor the colluding bytes within the operator's anchor interval, raising the conjunction-cost of the attack. Operators MAY add further anchors (e.g. a Deployer-operated transparency log or a witness service) without changing the wire format defined here.

* Use side-by-side chain comparison under regulator subpoena. The audit-trail alternative semantics established for SEC 17a-4 recordkeeping (see Section 7.6.1) and the post-market surveillance regime of EU AI Act Articles 12 and 26 (see Section 6.1 and Section 6.2) authorise the regulator to compel both A's and B's Audit Packs and to reconstruct the relay by joining on action_ref per Section 5.1.5. counterparty_binding reduces the regulator's workload from "compare both chains and detect divergence" to "verify B's bound digest against A's stored envelope"; the underlying subpoena-and-compare workflow remains the regulator's ultimate authority and remains operative when the binding is unverifiable.

* Document M's relay logs out-of-band. Where the intermediary M is identifiable (a named MCP server, message bus, orchestrator, or relay), the operator SHOULD require M to produce signed relay logs covering the time window of the Action and SHOULD submit those logs to the same Audit Pack production layer as A's and B's chains. Out-of-band relay logs do not require a wire-format change in this profile; they are operational evidence that complements counterparty_binding rather than replacing it.

The worst-case latency to detection for an M-only compromise (not M-B collusion) is bounded by the anchor interval recommended in Section 10.1: 24 hours by default, one hour for Deployers operating under [DORA] Article 17, 23 NYCRR 500.17, or [CIRCIA]. After the next anchor commits, the divergence between A's anchored chain segment and the digest carried in B's counterparty_binding is permanently recoverable from the anchored evidence alone, without trust in M.

11. IANA Considerations

This document requests two new IANA registries to support stable, machine-checkable extensions to the Compliance Receipt format.

11.1. Compliance Receipt Extension Fields Registry

IANA is requested to create a new registry titled "Compliance Receipt Extension Fields" under a new "Compliance Receipts" registry group.

This registry covers both signed-payload fields and envelope-level fields (siblings of payload and signature); each entry's Description identifies which.

Each entry contains:

* Field Name: a JSON object key, lowercase ASCII letters, digits, and underscore.

* Description: a one-line summary of the field's purpose.

* Reference: the document that defines the field's semantics.

* Vocabulary: a URL or registry pointer for the controlled vocabulary that field values are drawn from, or "free-form" if none.

The registration policy is Specification Required, per [RFC8126]. The Designated Expert(s) SHOULD verify that the field name does not collide with any field defined by [ACTA-RECEIPTS], that the Reference is a stable, dereferenceable specification, and that the Vocabulary is documented sufficiently for an independent verifier to validate values.

Initial registry contents (each entry labelled with Scope to disambiguate signed-payload fields from envelope-level fields per the registry description above):

* risk_class - Scope: signed-payload - Risk classification term under the Deployer's risk management documentation - This document - Vocabulary referenced in Audit Pack metadata.

* incident_class - Scope: signed-payload - Incident classification term spanning DORA Article 18(1) (with further specification in [REG-2024-1772] and the canonical reporting enumeration of Annex II field 3.23 of [REG-2025-302]), 23 NYCRR 500.1 Cybersecurity Event/Incident, [CIRCIA] Covered Cyber Incident, and HIPAA security incident under 45 CFR 164.304 - This document - Audit Pack metadata.

* counterparty_binding - Scope: signed-payload - Signed-payload object carrying a base64-encoded SHA-256 digest (envelope_hash) of a peer agent's full signed envelope including signature bytes, a

resolvable opaque locator (receipt_ref), an OPTIONAL expected-acknowledger identifier (expect_ack_from), and an OPTIONAL operational transport_label; see Section 5.6 for the full member set and the digest-scope rule - This document - Member vocabulary defined in Section 5.6.1; digest algorithm is SHA-256 per [ACTA-RECEIPTS] with base64 encoding per [RFC4648].

* result_digest - Scope: signed-payload - Object of the upstream payload_digest shape (hash, size, OPTIONAL preview) carrying a SHA-256 digest of the downstream Action’s result body; defined in Section 5.7 - This document - Digest algorithm is SHA-256 with hex encoding under the sha256:<64 hex> form.

* expires_at - Scope: signed-payload - ISO 8601 timestamp with explicit timezone declaring the wall-clock time after which the decision result is stale and not safe to replay; defined in Section 5.7 - This document - Free-form ISO 8601 string.

* nonce - Scope: signed-payload - Producer-generated string unique across the producer’s emission stream for the lifetime of kid; defined in Section 5.7 - This document - Free-form (base64url-encoded random value of at least 16 bytes, or a structured identifier such as UUIDv4, UUIDv7, ULID).

* tool_fingerprint - Scope: signed-payload - JSON string formatted sha256:<64 hex> over the JCS-canonical declaration of the tool that produced the Action; defined in Section 5.7 - This document - Digest algorithm is SHA-256 per [RFC8785] canonicalization.

* config_manifest_digest - Scope: signed-payload - JSON string formatted sha256:<64 hex> over the canonical bytes of the producer’s configuration manifest in effect at signing time; defined in Section 5.7 - This document - Manifest content is operator-defined; canonicalization rule is operator-declared in the Audit Pack manifest entry.

* cve_inventory_digest - Scope: signed-payload - JSON string formatted sha256:<64 hex> over the canonical bytes of the producer’s CVE inventory at signing time; defined in Section 5.7 - This document - Inventory content lists CVE identifiers per the producer’s accepted-residual rationale.

* executable_hash - Scope: signed-payload - JSON string formatted sha256:<64 hex> over the canonical bytes of the executable that invoked the Action (OCI image manifest digest for container producers, on-disk SHA-256 for non-container executables); defined in Section 5.8 - This document - Digest algorithm is SHA-256.

* sbom_digest - Scope: signed-payload - JSON string formatted sha256:<64 hex> over the canonical bytes of the CycloneDX or SPDX SBOM document covering the executing image; defined in Section 5.8 - This document - SBOM format and version declared in the Audit Pack manifest entry.

* slsa_provenance_pointer - Scope: signed-payload - JSON string carrying an https URL resolving to the SLSA provenance attestation envelope for the build of the executable identified by executable_hash; defined in Section 5.8 - This document - Target SHOULD be SLSA Provenance v1.0 in-toto statement form.

* supply_chain_pointer - Scope: signed-payload - JSON string carrying an https URL resolving to a transparency-log entry (in-toto, Sigstore, or Rekor) covering the build of the executable identified by executable_hash; defined in Section 5.8 - This document - Verifier acceptance is SHOULD across the three log formats.

* anchors - Scope: envelope-level - Envelope-level array of timestamp / transparency-log anchors covering the signed envelope; entries carry a REQUIRED type discriminator (rfc3161 or opentimestamps) and a REQUIRED value field, plus OPTIONAL informational members status (anchored / pending / failed) and bitcoin_block (string Bitcoin block hash for upgraded OpenTimestamps entries); full schema is defined in Section 5.4 - This document - Anchor type vocabulary: rfc3161 per [RFC3161], opentimestamps per [OPENTIMESTAMPS].

* witness_policy - Scope: envelope-level - Envelope-level object declaring an N-of-M durable-anchoring quorum over the anchors array; carries a REQUIRED integer required in the closed range [1, length of witnesses] and a REQUIRED non-empty array witnesses whose distinct values are a subset of {rfc3161, opentimestamps} (Rekor and other transparency-log pointers are NOT witness types and MUST be rejected). The receipt reaches the quorum-met state only when at least required distinct witness types each hold a verifiable inclusion proof; a producer MUST NOT assert durable anchoring unless that holds, per the false-attestation rule of Section 5.4 - This document - Witness type vocabulary: rfc3161 per [RFC3161], opentimestamps per [OPENTIMESTAMPS]; inclusion-proof prior art per [DRAFT-IETF-SCITT] and [RFC6962].

* mitre_techniques - Scope: signed-payload - JSON array of MITRE ATT&CK technique identifiers (for example T1059, T1078) self-declared by the producer; not verifier-checked by the issuing platform; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced by id from the MITRE ATT&CK enterprise matrix.

* mitre_atlas - Scope: signed-payload - JSON array of MITRE ATLAS identifiers (for example AML.T0051) covering AI-system-specific adversary techniques; self-declared; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced by id from the MITRE ATLAS catalogue.

* owasp_llm_top10 - Scope: signed-payload - JSON array of OWASP Top 10 for LLM Applications identifiers (for example LLM01, LLM02); self-declared; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced by id from the OWASP Top 10 for LLM Applications publication.

* nist_ai_rmf - Scope: signed-payload - JSON array of NIST AI Risk Management Framework function identifiers and subcategories (for example GOVERN-1.1, MEASURE-2.7); self-declared; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced from NIST AI RMF 1.0.

* iso_42001 - Scope: signed-payload - JSON array of ISO/IEC 42001:2023 control identifiers (for example A.6.2.6); self-declared; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced from ISO/IEC 42001:2023.

* eu_ai_act_articles - Scope: signed-payload - JSON array of EU AI Act article identifiers (for example Article-12, Article-15); self-declared; flips framework_mappings_self_declared to true when populated; defined in Section 5.10 - This document - Vocabulary referenced from [EU-AI-ACT].

* rfc3161_timestamp - Scope: signed-payload - JSON string carrying a base64-encoded RFC 3161 TimeStampResp (DER) supplied by the producer at signing time and preserved verbatim on the receipt for offline TSA chain verification independent of any platform-issued anchors; payload entry is an opaque caller-supplied token, not the per-receipt anchor produced by the platform; defined in Section 5.10 - This document - Bytes are an RFC 3161 TimeStampResp per [RFC3161]; base64 encoding per [RFC4648].

* framework_mappings_self_declared - Scope: signed-payload - JSON boolean false-attestation guard set by the issuing platform to true whenever any of mitre_techniques, mitre_atlas, owasp_llm_top10, nist_ai_rmf, iso_42001, or eu_ai_act_articles is populated; a producer-supplied value of false alongside a populated taxonomy field MUST be overridden by the issuing platform; defined in Section 5.10 - This document - Boolean.

* authorized_under_mandate - Scope: signed-payload - Server-built object recording a self-declared authorizing mandate the Action was signed under; carries REQUIRED mandate_id, REQUIRED issuer_id, a REQUIRED scope_digest formatted sha256:<64 hex> over the mandate's authorized-action-types scope, and a REQUIRED verified boolean whose true value asserts self-declared issuer authority (the same trust level as framework_mappings_self_declared, never issuing-platform-verified third-party authorization); a present-but-malformed object is rejected by the false-attestation guard at signing time; defined in Section 5.9 - This document - Member vocabulary defined in Section 5.9; scope_digest digest algorithm is SHA-256.

* controls_evaluated - Scope: signed-payload - Server-built object enumerating the enforcement controls that genuinely fired on this sign plus the allow result; member keys are a closed set (emergency_halt, delegation_scope, quorum, mandate, policy, content_scan, result) and an unknown key is rejected; a key is present only when its control ran (omission-over-false attestation), quorum requires fired=true plus a 64-hex attestation_hash, and a policy member asserting evaluation requires matched_count at least 1; a caller-supplied value is dropped before signing; defined in Section 5.9 - This document - Control-key vocabulary defined in Section 5.9.

This document additionally requests that IANA register counterparty_binding as a new claim in the CBOR Web Token (CWT) Claims registry established by [RFC8392], with this document as the reference and the semantics defined in Section 5.6. The requested CWT claim key is TBD by IANA; this document proposes allocation from the IANA First Come First Served range (claim keys greater than or equal to -65536 and less than -256, or greater than or equal to 65536, per [RFC8392] Section 9.1) so as not to consume Expert Review or Standards Action space. The JSON-form claim name is the literal string counterparty_binding as registered above in the Compliance Receipt Extension Fields Registry.

11.2. Compliance Receipt Type Namespaces Registry

IANA is requested to create a new registry titled "Compliance Receipt Type Namespaces" under the same "Compliance Receipts" registry group.

Each entry contains:

* Namespace: a colon-separated identifier prefix used as a value of the type field, lowercase ASCII letters, digits, hyphen, underscore, and colon.

* Description: a one-line summary of the receipt category.

* Reference: the document that defines the namespace.

The registration policy is Specification Required, per [RFC8126]. The Designated Expert(s) SHOULD verify that the namespace does not collide with any namespace already registered or any namespace reserved by [ACTA-RECEIPTS], and that the Reference is a stable specification.

Initial registry contents:

* protectmcp:acknowledgment - A receipt emitted by the B-party in a counterparty_binding pair, confirming receipt of the A-party action and carrying a digest of the bound envelope per Section 5.6 - This document.

* protectmcp:decision - A receipt recording a policy evaluation outcome (allow, deny, rate_limit) for an MCP-mediated tool call where a policy was actually evaluated; observation is reserved to protectmcp:lifecycle and protectmcp:observation per Section 5.2 - This document.

* protectmcp:restraint - A receipt recording the application or release of a restraint on an agent (e.g., quota, rate limit, sandbox tightening) - This document.

* protectmcp:lifecycle - A receipt recording an agent or system lifecycle event (e.g., configuration change, key rotation, oversight review, or a decision=observation record indicating an Action was signed without policy evaluation per Section 5.2) - This document.

* protectmcp:lifecycle:configuration_change - A receipt recording a configuration change to an agent or producing system, including changes that disable or re-enable receipt generation (see Section 6.1.1); a registered sub-namespace under protectmcp:lifecycle that the reference cloud implementation emits today - This document.

* protectmcp:observation - A receipt recording passive telemetry about an Action that was signed without a policy evaluation, emitted under one of the capture topologies catalogued in Appendix "Appendix - Capture Topologies for Compliance Receipt Emission" (typically network_proxy, browser_extension, ebpf_observer, or mcp_proxy) where the originating application could not call the receipt-emitting SDK directly - This document.

* protectmcp:observation:result_bound - A sub-namespace under protectmcp:observation for a follow-up observation receipt that carries a result_digest (Section 5.7) binding the byte-equality of a downstream Action’s result to the originating protectmcp:decision receipt identified by action_ref; the reference cloud implementation emits this type for tool calls whose downstream result is regulator-relevant (LLM completions under EU AI Act Article 12, audit-log entries under HIPAA 164.312(b), broker-dealer communications under SEC 17a-4) - This document.

12. Related Work

This section catalogues parallel proposals that overlap the problem space of signed records for AI-agent actions. The intent is to position this profile within the broader Independent Submissions and individual drafts landscape so that an implementer can choose the surface that matches the trust model and the regulator that the implementer is bound by. The citations follow the RFC 7322 form and are informative; this section places no normative constraint on a Compliance Receipt implementation.

* [DRAFT-SHARIF-APKI] defines a certificate-based Public Key Infrastructure for autonomous AI agents, with X.509 certificate profiles, naming conventions, and revocation considerations. Its scope is identity issuance and trust-root anchoring for agent principals; it does not define a per-action receipt format. This profile is complementary to [DRAFT-SHARIF-APKI]: an implementer that anchors agent identity through APKI can express the resulting kid and issuer_id values in Compliance Receipts under Section 5.1.3, with the trust-anchor metadata of the Audit Pack resolving through the APKI certificate chain.

* [DRAFT-SHARIF-AML] defines cryptographic attestation across the AI model lifecycle, from training-data provenance to inference-time output binding. Its scope is the model-build and model-evaluation surface, addressing what evidence a model producer or model consumer SHOULD retain about the model's lineage and behaviour. This profile is complementary at the per-action layer: a Deployer that consumes a model whose lifecycle attestations are produced under [DRAFT-SHARIF-AML] can reference those attestation digests in Compliance Receipts via config_manifest_digest (Section 5.7) or via the build-provenance four-tuple of Section 5.8 where the model is delivered as a containerized executable.

* [PIPELOCK-ER2] proposes an alternative receipt format for AI agent actions under the EvidenceReceipt v2 schema. The Pipelock format is published as a public reference outside the IETF process and addresses overlapping evidence-class requirements with a different envelope, a different canonicalization choice, and a different anchoring posture. This profile and the Pipelock format are bidding for the same regulator audience under different design tradeoffs; an implementer that already deploys EvidenceReceipt v2 receipts can map the field set across the two surfaces, though byte-equality across the formats is not preserved (the canonicalization rules differ) and a cross-format verifier would have to recompute digests under each format's rule.

* [DRAFT-IETF-SCITT] defines the Supply Chain Integrity, Transparency, and Trust architecture: an append-only Transparency Service that admits signed statements about an artefact and returns a receipt proving inclusion in the log, with verification reducing to a Merkle inclusion proof in the tradition of [RFC6962] Certificate Transparency. SCITT and RFC 6962 are the canonical inclusion-proof prior art that this profile's witness_policy (Section 5.4) generalizes from a single log to an N-of-M quorum over heterogeneous durable-anchoring witnesses. This profile is complementary to SCITT rather than competitive with it: a SCITT Transparent Statement MAY carry a Compliance Receipt as its signed payload, so that a Deployer operating a SCITT Transparency Service obtains a SCITT receipt whose inclusion proof can be named as one witness in a Compliance Receipt's witness_policy, while the Compliance Receipt continues to carry the regulator-facing field bindings that SCITT's artefact-neutral envelope does not define. This closes the append-only-log gap deferred under Section 10.12.

* [A2A] defines the Agent2Agent protocol, a vendor-neutral horizontal protocol for communication between independent agents, hosted as a Linux Foundation project. A2A defines an Agent Card, an extension mechanism (data-only, profile, method, and state-machine extension classes declared inside the Agent Card

capabilities and activated by clients through a request header), and an Agent Card signature that carries a detached JSON Web Signature over the Agent Card document itself. A2A is complementary to this profile rather than competitive with it: A2A's Agent Card signature attests the integrity of the card, not the execution of a per-action authorization decision, and A2A is explicitly method-agnostic on identity verification. A Deployer can reference a Compliance Receipt from an A2A Agent Card through a verifier-issued attestation pointer (a resolvable URL, a content hash over the referenced receipt, and the verifier identity expressed as a decentralized identifier), so that the Compliance Receipt's issuer_id (Section 5.1.3) is the receipt subject and the per-action regulator-facing bindings of Sections 5 and 6 compose on top of A2A's identity layer without either layer redefining the other.

* [A2A-IDF] is the Agent Identity Verification and Trust Framework proposal under the A2A protocol's contribution process, defining tiered identity verification levels (self-asserted, domain-verified, organization-verified), Agent Card signing for production agents, revocation endpoints, delegation-chain verification, and per-message signing. Its scope is the identity and trust-anchoring layer beneath per-action receipts; it does not define a regulator-facing per-action receipt format, and its post-quantum cryptography work is scheduled for a later cycle. This profile is complementary to [A2A-IDF] at the per-action layer: an implementer that anchors agent identity through an A2A-IDF verification level can express the resulting identity as the issuer_id and kid of Compliance Receipts under Section 5.1.3, while this profile already signs receipts with the ML-DSA-65 post-quantum signature algorithm of [FIPS204] that the surrounding agent-identity ecosystem largely defers.

* [DRAFT-HOPLEY-X402] is the closest parallel receipt format: it defines a Compliance Receipt minted at admission time under a payment-gate remote procedure call, carrying an ALLOW, REFER, or DENY verdict bound to a payment mandate. The format canonicalizes under [RFC8785] JCS, digests under SHA-256, and links receipts in a monotonic hash chain (position plus content hash plus previous-hash), in the same family of design primitives as this profile (JCS canonicalization, SHA-256 digests, hash-chain linkage under Section 5.3, and a categorical decision under Section 5.2 bound to an authorizing mandate under Section 5.9). The two drafts differ in scope and trust model. [DRAFT-HOPLEY-X402] is narrowed to payment anti-money-laundering and sanctions-screening admission decisions, is bound to a payment-settlement substrate, and its REFER verdict carries a jurisdiction-specific suspicious-activity-reporting meaning. This profile is the broader superset: it

covers general agent-action receipts across regulatory regimes rather than payment screening alone (Sections 5 and 6), it is signed by a neutral third party rather than by the screening party itself (the unaffiliated-custodian posture the recordkeeping bindings of Sections 5 and 6 anchor), it carries the organization-wide enforcement controls enumerated in controls_evaluated (Section 5.9) rather than a single admission verdict, and it signs with the ML-DSA-65 post-quantum algorithm of [FIPS204]. An implementer deploying both can map the verdict and mandate-binding fields across the two surfaces, though byte-equality is not preserved because the envelope shapes differ.

This section is non-exhaustive. The Independent Submissions stream and the individual-draft search surface of the IETF Datatracker host additional proposals whose scope intersects the Compliance Receipt problem space; an implementer evaluating the field is encouraged to re-query the Datatracker against the keywords "agent", "receipt", "attestation", and "audit trail" at the time of implementation. The author welcomes pull requests against the source repository of this draft naming additional active parallel work.

13. Acknowledgements

The author thanks Tom Farley for [ACTA-RECEIPTS], on which this profile is built. This profile would not exist without the field catalogue and envelope structure that the upstream draft defines. The author also thanks the Asqav community for review of early drafts.

14. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", FIPS 204, DOI 10.6028/NIST.FIPS.204, 13 August 2024, <https://csrc.nist.gov/pubs/fips/204/final>.

[RFC5816] Santesson, S. and N. Pope, "ESSCertIDv2 Update for RFC 3161", RFC 5816, DOI 10.17487/RFC5816, April 2010, <https://www.rfc-editor.org/info/rfc5816>.

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[OPENTIMESTAMPS] OpenTimestamps, "OpenTimestamps Server", September 2016, <https://github.com/opentimestamps/opentimestamps-server>.

[ACTA-RECEIPTS] Farley, T., "Signed Decision Receipts for Machine-to-Machine Access Control", Work in Progress, Internet-Draft, draft-farley-acta-signed-receipts-01, 25 April 2026, <https://datatracker.ietf.org/doc/html/draft-farley-acta-signed-receipts-01>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

[ISO17442] ISO, "Financial services - Legal entity identifier (LEI) - Part 1: Assignment", ISO 17442-1:2020, August 2020, <https://www.iso.org/standard/78829.html>.

[W3C-DID] W3C, "Decentralized Identifiers (DIDs) v1.0", 19 July 2022, <https://www.w3.org/TR/did-1.0/>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[NIST-GENAI-PROFILE] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile", NIST AI 600-1, DOI 10.6028/NIST.AI.600-1, 26 July 2024, <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf>.

15. Informative References

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/info/rfc5705>.

[RFC9266] Whited, S., "Channel Bindings for TLS 1.3", RFC 9266, DOI 10.17487/RFC9266, July 2022, <https://www.rfc-editor.org/info/rfc9266>.

[RFC9421] Backman, A., Ed., Richer, J., Ed., and M. Sporny, "HTTP Message Signatures", RFC 9421, DOI 10.17487/RFC9421, February 2024, <https://www.rfc-editor.org/info/rfc9421>.

[ISO8601-2] ISO, "Date and time - Representations for information interchange - Part 2: Extensions", ISO 8601-2:2019, February 2019, <https://www.iso.org/standard/70908.html>.

[EU-AI-ACT] European Parliament and Council, "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance)", 12 July 2024, <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>.

[DORA] European Parliament and Council, "Regulation (EU) 2022/2554 of the European Parliament and of the Council of 14 December 2022 on digital operational resilience for the financial sector and amending Regulations (EC) No 1060/2009, (EU) No 648/2012, (EU) No 600/2014, (EU) No 909/2014 and (EU) 2016/1011 (Text with EEA relevance)", 27 December 2022, <https://eur-lex.europa.eu/eli/reg/2022/2554/oj>.

[REG-2025-301] European Commission, "Commission Delegated Regulation (EU) 2025/301 of 23 October 2024 supplementing Regulation (EU) 2022/2554 of the European Parliament and of the Council with regard to regulatory technical standards specifying the content, timelines and templates on the reporting of major ICT-related incidents and significant cyber threats (Text with EEA relevance)", 20 February 2025, <https://eur-lex.europa.eu/eli/reg_del/2025/301/oj>.

[REG-2025-302] European Commission, "Commission Implementing Regulation (EU) 2025/302 of 23 October 2024 laying down implementing technical standards for the application of Regulation (EU) 2022/2554 of the European Parliament and of the Council with regard to the standard forms, templates, and procedures for financial entities to report a major ICT-related incident and to notify a significant cyber threat (Text with EEA relevance)", 20 February 2025, <https://eur-lex.europa.eu/eli/reg_impl/2025/302/oj>.

[REG-2024-1772] European Commission, "Commission Delegated Regulation (EU) 2024/1772 of 13 March 2024 supplementing Regulation (EU) 2022/2554 of the European Parliament and of the Council with regard to regulatory technical standards specifying the criteria for the classification of ICT-related incidents and cyber threats, setting out materiality

thresholds and specifying the details of reports of major incidents (Text with EEA relevance)", 25 June 2024, <https://eur-lex.europa.eu/eli/reg_del/2024/1772/oj>.

[MIFID2] European Parliament and Council, "Directive 2014/65/EU of the European Parliament and of the Council of 15 May 2014 on markets in financial instruments and amending Directive 2002/92/EC and Directive 2011/61/EU (recast) (Text with EEA relevance)", 12 June 2014, <https://eur-lex.europa.eu/eli/dir/2014/65/oj>.

[REG-2017-565] European Commission, "Commission Delegated Regulation (EU) 2017/565 of 25 April 2016 supplementing Directive 2014/65/EU of the European Parliament and of the Council as regards organisational requirements and operating conditions for investment firms and defined terms for the purposes of that Directive (Text with EEA relevance)", 31 March 2017, <https://eur-lex.europa.eu/eli/reg_del/2017/565/oj>.

[AMLD] European Parliament and Council, "Directive (EU) 2015/849 of the European Parliament and of the Council of 20 May 2015 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing, amending Regulation (EU) No 648/2012 of the European Parliament and of the Council, and repealing Directive 2005/60/EC of the European Parliament and of the Council and Commission Directive 2006/70/EC (Text with EEA relevance)", 5 June 2015, <https://eur-lex.europa.eu/eli/dir/2015/849/oj>.

[AMLR] European Parliament and Council, "Regulation (EU) 2024/1624 of the European Parliament and of the Council of 31 May 2024 on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing (Text with EEA relevance)", 19 June 2024, <https://eur-lex.europa.eu/eli/reg/2024/1624/oj>.

[NIST-AI-RMF] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)", NIST AI 100-1, DOI 10.6028/NIST.AI.100-1, 26 January 2023, <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf>.

[COLORADO-AI-ACT]
State of Colorado, Seventy-Fourth General Assembly, "Senate Bill 24-205, Consumer Protections for Artificial Intelligence", 17 May 2024, <https://leg.colorado.gov/bills/sb24-205>.

[TEXAS-TRAIGA]
State of Texas, 89th Legislature, Regular Session, "House Bill 149, Texas Responsible Artificial Intelligence Governance Act", 22 June 2025, <https://capitol.texas.gov/BillLookup/History.aspx?LegSess=89R&Bill=HB149>.

[HIPAA-SECURITY]
United States Department of Health and Human Services, "HIPAA Security Rule, 45 CFR Part 164, Subpart C, Security Standards for the Protection of Electronic Protected Health Information", 20 February 2003, <https://www.ecfr.gov/current/title-45/subtitle-A/subchapter-C/part-164>.

[NYDFS-500]
New York State Department of Financial Services, "23 NYCRR Part 500, Cybersecurity Requirements for Financial Services Companies", 1 March 2017, <https://www.dfs.ny.gov/industry-guidance/cybersecurity>.

[SEC-17A-4]
United States Securities and Exchange Commission, "Electronic Recordkeeping Requirements for Broker-Dealers, Security-Based Swap Dealers, and Major Security-Based Swap Participants (Rule 17a-4 Amendments)", 3 November 2022, <https://www.federalregister.gov/documents/2022/11/03/2022-22670/electronic-recordkeeping-requirements-for-broker-dealers-security-based-swap-dealers-and-major>. Effective date January 3, 2023; compliance date for amendments to 17 CFR 240.17a-4 May 3, 2023.

[CIRCIA] United States Congress, "Cyber Incident Reporting for Critical Infrastructure Act of 2022, enacted as Division Y of the Consolidated Appropriations Act, 2022 (Public Law 117-103); statutory authority codified at 6 U.S.C. 681 et seq.", 15 March 2022, <https://www.congress.gov/117/plaws/publ103/PLAW-117publ103.pdf>. Public Law 117-103 was enacted on March 15, 2022. Implementing regulations are proceeding under a CISA notice of proposed rulemaking at 89 FR 23644 (April 4, 2024); the final rule is pending publication.

[DRAFT-SHARIF-APKI] Independent, "Agent Public Key Infrastructure (APKI): Certificate-Based Identity and Trust for Autonomous AI Agents", Work in Progress, Internet-Draft, draft-sharif-apki-agent-pki-00, 2025, <https://datatracker.ietf.org/doc/draft-sharif-apki-agent-pki/>. Active Independent Submission on the IETF Datatracker; defines a certificate-based PKI for autonomous AI agents. Cited under Section 12 as parallel work on the identity-anchoring surface complementary to this profile’s per-action receipt format.

[DRAFT-SHARIF-AML] Independent, "Cryptographic Attestation for AI Model Lifecycle: From Training Data to Inference Output", Work in Progress, Internet-Draft, draft-sharif-ai-model-lifecycle-attestation-00, 2025, <https://datatracker.ietf.org/doc/draft-sharif-ai-model-lifecycle-attestation/>. Active Independent Submission on the IETF Datatracker; defines cryptographic attestation across the AI model lifecycle from training- data provenance to inference-time output binding. Cited under Section 12 as parallel work on the model-build and model-evaluation surface complementary to this profile’s per-action receipt format.

[PIPELOCK-ER2] Pipelock, "EvidenceReceipt v2: An Alternative Receipt Format for AI Agent Actions", 2025, <https://pipelock.ai/evidence-receipt-v2>. Public reference published outside the IETF process. Cited under Section 12 as an alternative receipt-format proposal under different envelope, canonicalization, and anchoring choices; byte-equality across the two formats is not preserved.

[A2A] A2A Project, a Linux Foundation project, "Agent2Agent (A2A) Protocol", 2025, <https://github.com/a2aproject/A2A>. Vendor-neutral horizontal protocol for communication between independent agents, hosted as a Linux Foundation project. Defines an Agent Card, an extension mechanism, and a JSON Web Signature over the Agent Card document. Cited under Section 12 as parallel work on the agent- interoperation surface complementary to this profile's per-action receipt format; A2A is method-agnostic on identity verification and does not define a regulator-facing per-action receipt format.

[A2A-IDF] A2A Project contributors, "Agent Identity Verification and Trust Framework (A2A-IDF)", 2026, <https://github.com/a2aproject/A2A/issues/1497>. Identity and trust-framework proposal under the A2A protocol contribution process, defining tiered verification levels, Agent Card signing, revocation, and delegation-chain verification. Cited under Section 12 as parallel work on the identity-anchoring layer beneath this profile's per-action receipts; post-quantum cryptography is scheduled for a later cycle.

[DRAFT-HOPLEY-X402] Hopley, C., "x402 Compliance Receipt", Work in Progress, Internet-Draft, draft-hopley-x402-compliance-receipt-02, 25 May 2026, <https://datatracker.ietf.org/doc/draft-hopley-x402-compliance-receipt/>. Independent Submission, Informational. Defines a JCS-canonicalized, SHA-256 hash-chained Compliance Receipt carrying an ALLOW, REFER, or DENY admission verdict bound to a payment mandate under the x402 payment substrate. Cited under Section 12 as the closest parallel receipt format; this profile is the broader superset (general agent-action receipts, neutral third-party signing, organization-wide control attestation, ML-DSA-65) and that draft is narrowed to payment anti- money-laundering and sanctions-screening admission decisions.

[DRAFT-IETF-SCITT] IETF SCITT Working Group, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture, 2025, <https://datatracker.ietf.org/doc/draft-ietf-scitt-architecture/>. IETF SCITT Working Group draft defining an append-only Transparency Service that issues inclusion-proof receipts over signed statements. Cited under

Section 12 as canonical inclusion-proof prior art complementary to this profile's witness_policy; a SCITT Transparent Statement can carry a Compliance Receipt as its payload.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>. Defines the Merkle-tree inclusion-proof model for an append-only public log. Cited under Section 12 as the foundational inclusion-proof prior art that this profile's witness_policy generalizes to an N-of-M quorum over heterogeneous durable-anchoring witnesses.

Worked Example (Informative)

This appendix illustrates a Compliance Receipt that satisfies the EU AI Act Article 26 binding for a tool invocation by a High-Risk AI System deployed by a Financial Entity. The wire shape applies identically to United States bindings; the only differences are the values placed in issuer_id (LEI, EIN, or CIK depending on the regime) and in the risk_class and incident_class vocabularies referenced in the Audit Pack manifest. Field values are abbreviated for readability and are not cryptographically valid. The example shows a mid-chain receipt; a chain-genesis receipt would carry a previousReceiptHash of 64 zero hex characters per Section 5.3.

The worked example below is illustrative. The digest-valued fields (action_ref, policy_digest, the hash member of payload_digest, and previousReceiptHash) are shown abbreviated with an internal ellipsis so each line fits the column width; a real receipt carries the full 64-character lowercase hex digest. The keys are shown in human-readable order rather than JCS-canonical lexicographic order; the JCS-canonical bytes used as input to SHA-256 reorder keys lexicographically (so the on-the-wire byte order for hashing is action_ref, decision, issued_at, issuer_id, iteration_id, payload_digest, policy_digest, previousReceiptHash, reason, risk_class, sandbox_state, tool_name, type, with each nested object's keys also lexicographically sorted, per [RFC8785]). The sig value, the anchors value entries, and the hash hex values are deterministic placeholders chosen for shape rather than cryptographic validity. Implementations MUST NOT replay or trust this example as a real receipt; the example is not signed by any allocated issuer_id, is not anchored against any TSA or OpenTimestamps calendar, and does not chain into any retained predecessor.

```
{
  "payload": {
    "type": "protectmcp:decision",
    "issued_at": "2026-05-04T09:14:22.118Z",
    "issuer_id": "00000000000000000098",
    "action_ref": "c1f3a09a4d2e7f6b8c5a91e3d7b04f2a...3c5e7f9d",
    "tool_name": "deploy",
    "iteration_id": "task-2026-05-04-01a3",
    "decision": "allow",
    "reason": "policy:within_limits",
    "policy_digest": "sha256:7b214e8c3d9f4a2b1e6c8f5a...1f3a5d7b",
    "sandbox_state": "enabled",
    "payload_digest": {
      "hash": "0a44d2c8e3f5b7a9d1c4e6f8b2a5d7c9...e7f9b2a4",
      "size": 1024
    },
    "previousReceiptHash": "f80c11a3b5d7e9c2f4a6b8d1...b8d1e3c5",
    "risk_class": "deployer:financial:medium"
  },
  "signature": {
    "alg": "EdDSA",
    "kid": "00000000000000000098",
    "sig": "..."
  },
  "anchors": [
    {
      "type": "rfc3161",
      "value": "..."
    },
    {
      "type": "opentimestamps",
      "value": "..."
    }
  ]
}
```


The above receipt satisfies the Article 26 binding because:

* issuer_id is a 20-character ISO 17442 Legal Entity Identifier (LEI) that resolves through the trust anchor metadata in the Audit Pack to the named Deployer;

* policy_digest resolves to a retained policy artefact;

* sandbox_state is enabled, satisfying the High-Risk system constraint of Section 5.1.6;

* previousReceiptHash links the receipt into the chain per Section 5.3;

* both an [RFC3161] anchor and an [OPENTIMESTAMPS] anchor are present per Section 5.4.

Under the DORA Article 17 binding a Compliance Verifier additionally checks the longest applicable sectoral retention floor (1827 days as the default, per Section 6.3.4) and, where present, that incident_class flattens to the canonical vocabulary referenced in Section 5.5 (resolved from [REG-2025-302] Annex II field 3.23 directly). Under the United States bindings of Section 6 the same verifier additionally checks the longest applicable retention floor (2192 days as the default for receipts under Section 7.4 or Section 7.6) and, where the Deployer is a NYDFS Covered Entity, a HIPAA Covered Entity, or a CIRCIA Covered Entity, that incident_class resolves to the applicable canonical category for each in-scope regime.

Change Log

[RFC Editor: please remove this appendix and its subsections before publication.]

Changes in draft -05

Build-provenance and result-bound extension revision. This revision registers vocabulary that the reference cloud implementation, the asqav-sdk Python and TypeScript halves, and the published /.well-known/governance.json already emit and accept, bringing the IETF surface back into parity with deployed reality.

* New normative Section 5.7 defines six OPTIONAL signed-payload extension fields: result_digest (object of the upstream payload_digest shape carrying the downstream Action’s result-body SHA-256), expires_at (ISO 8601 staleness bound additive to the 300-second forward-skew rule of Section 5.1.2), nonce (producer-unique replay-resistance value), tool_fingerprint (JCS-canonicalized tool-declaration SHA-256), config_manifest_digest (operator-defined configuration manifest SHA-256), and cve_inventory_digest (CVE inventory SHA-256 in effect at signing time). The six fields are type-agnostic; result_digest typically appears on receipts of the newly registered protectmcp:observation:result_bound namespace.

* New normative Section 5.8 defines a four-tuple of OPTIONAL signed-payload extension fields binding the executing build to the receipt: executable_hash (OCI image manifest digest for container

producers, on-disk SHA-256 for non-container executables), sbom_digest (CycloneDX or SPDX SBOM SHA-256), slsa_provenance_pointer (https URL to the SLSA Provenance v1.0 in-toto attestation envelope), and supply_chain_pointer (https URL to an in-toto, Sigstore, or Rekor transparency-log entry covering the build). The four fields form a layered subsumption set; an implementation MAY emit any subset.

* Section 11.1 Initial registry contents are extended with the ten new fields registered in this revision (result_digest, expires_at, nonce, tool_fingerprint, config_manifest_digest, cve_inventory_digest, executable_hash, sbom_digest, slsa_provenance_pointer, supply_chain_pointer), each labelled Scope: signed-payload. The four pre-existing entries (risk_class, incident_class, counterparty_binding, anchors) are unchanged.

* Section 11.2 Initial registry contents are extended with protectmcp:observation (a passive-telemetry receipt emitted under one of the capture topologies of Appendix "Appendix - Capture Topologies for Compliance Receipt Emission" when the originating application could not call the SDK directly) and protectmcp:observation:result_bound (a follow-up sub-namespace under protectmcp:observation for receipts carrying result_digest bindings to an originating protectmcp:decision receipt via action_ref).

* The abstract and Section 1.1 overview text are rewritten to enumerate the full extension-field set in five groupings (regulatory classification; cross-agent envelope binding; per-action freshness and integrity plus build-provenance; server-built enforcement attestation; and self-declared threat-framework taxonomy) rather than the legacy "three extension fields" phrasing carried forward from -02 through -04. The wire shape, the signature scope, the canonicalization rule, the hash chain, and the anchoring rules of earlier revisions are unchanged.

* New informative Section 12 cites parallel proposals under the Independent Submissions and individual-draft surfaces of the IETF Datatracker: [DRAFT-SHARIF-APKI] (certificate-based PKI for autonomous AI agents), [DRAFT-SHARIF-AML] (cryptographic attestation across the AI model lifecycle), and [PIPELOCK-ER2] (the Pipelock EvidenceReceipt v2 alternative receipt-format proposal, published as a public reference outside the IETF process). The section is non-exhaustive and is informative; this profile places no normative constraint on a Compliance Receipt implementation by virtue of citing parallel work.

* New informative references: [DRAFT-SHARIF-APKI], [DRAFT-SHARIF-AML], [PIPELOCK-ER2].

* The four-tuple of build-provenance extensions in Section 5.8 is the wire-level surface of the supply-chain notary use case; Deployers under [EU-AI-ACT] Article 12 read together with [DORA] Article 17, under 45 CFR 164.312(b) of [HIPAA-SECURITY], and under [SEC-17A-4] read with [NYDFS-500] 23 NYCRR 500.6, SHOULD emit at least executable_hash on every protectmcp:decision receipt. The wording is hortatory; the underlying regime obligations remain the load-bearing requirement.

* No changes to the signing algorithms, the canonicalization transformation (JCS), the hash-chain scope corrected in Appendix "Changes in draft -04", the anchor type set, the retention floors of Sections 5 and 6, the Audit Pack manifest fields, or the verifier reporting fields. A -04 receipt that omits all ten new extensions is a conformant -05 receipt. A -05 receipt that carries any of the ten new extensions remains a conformant [ACTA-RECEIPTS] receipt under the upstream extension semantics of Section 4.2 (verifiers that do not implement the extension ignore it).

* Threat-framework taxonomy subsumption. New normative Section 5.10 defines, and Section 11.1 Initial registry contents are further extended with, eight additional signed-payload entries that the reference cloud implementation, both SDK halves, and the published /.well-known/governance.json emit and accept: six caller-supplied taxonomy lists (mitre_techniques, mitre_atlas, owasp_llm_top10, nist_ai_rmf, iso_42001, eu_ai_act_articles), one caller-supplied opaque token (rfc3161_timestamp, a base64-encoded TimeStampResp per [RFC3161] preserved verbatim on the receipt for offline TSA chain verification independent of any platform-issued anchors), and one platform-set false-attestation guard boolean (framework_mappings_self_declared, set to true by the issuing platform whenever any of the six taxonomy lists is populated; a producer-supplied false alongside a populated taxonomy field MUST be overridden). The taxonomy fields are not platform-verified; the guard exists so verifiers can tell self-declared classifications apart from platform-verified ones. Pre-existing entries are unchanged. The total Initial registry contents now enumerate twenty-two signed-payload-or-envelope-level fields.

* Durable-anchoring quorum. Section 5.4 is extended with a normative OPTIONAL envelope-level witness_policy member (a sibling of payload, signature, and anchors) that the reference cloud implementation, both SDK halves, and the published /.well-known/governance.json emit and accept. witness_policy declares an N-of-M

quorum over the anchors array: required (integer in [1, length of witnesses]) and witnesses (non-empty array, distinct subset of {rfc3161, opentimestamps}; Rekor and other transparency-log pointers are rejected). The quorum-met state (reported by the reference implementation as witness_quorum_met) is reached only when at least required distinct witness types each hold a verifiable inclusion proof; a receipt MUST NOT claim durable anchoring otherwise, extending the false-attestation principle to the anchoring quorum. Section 11.1 Initial registry contents are extended with the witness_policy entry, labelled Scope: envelope-level, bringing the total to twenty-three signed-payload-or-envelope-level fields. The baseline single-anchor requirement of Section 5.4 is unchanged and continues to apply to every Compliance Receipt regardless of whether witness_policy is present.

* Section 12 is extended with a cite of [DRAFT-IETF-SCITT] (IETF SCITT architecture) and [RFC6962] (Certificate Transparency Merkle inclusion) as the canonical inclusion-proof prior art that witness_policy generalizes; SCITT is framed as complementary (a SCITT Transparent Statement can carry a Compliance Receipt as its payload), closing the append-only-log gap deferred under Section 10.12. New informative references: [DRAFT-IETF-SCITT], [RFC6962].

* Enforcement-attestation extensions. New normative Section 5.9 defines two OPTIONAL server-built signed-payload extension fields that the reference cloud implementation and the published /.well-known/governance.json emit: authorized_under_mandate (an object recording a self-declared authorizing mandate with mandate_id, issuer_id, a scope_digest over the mandate’s authorized-action-types scope, and a verified boolean whose true value asserts self-declared issuer authority, never platform-verified third-party authorization; the binding is evaluated against the issuing platform’s clock and scoped to action types, with no value cap and no counterparty restriction) and controls_evaluated (a server-built enumeration of which enforcement controls genuinely fired on the sign, drawn from the closed key set emergency_halt, delegation_scope, quorum, mandate, policy, content_scan, result, where an absent key means the control did not run rather than that it passed silently). Both fields are server-built, never request inputs; a caller-supplied value is dropped before signing. Each field carries a false-attestation guard that rejects a present-but-malformed attestation at signing time. Section 11.1 Initial registry contents are extended with the two new entries, each labelled Scope: signed-payload, bringing the total to twenty-five signed-payload-or-envelope-level fields.

* Section 12 is further extended with a cite of [A2A] (the Agent2Agent protocol, a Linux Foundation project with an Agent Card, an extension mechanism, and an Agent Card JSON Web Signature), [A2A-IDF] (the Agent Identity Verification and Trust Framework proposal under the A2A contribution process, framed as the identity-anchoring layer beneath this profile's per-action receipts), and [DRAFT-HOPLEY-X402] (the closest parallel receipt format: a JCS-canonicalized, SHA-256 hash-chained Compliance Receipt with an ALLOW, REFER, or DENY admission verdict bound to a payment mandate under the x402 substrate). This profile is framed as the broader superset of the x402 format: general agent-action receipts across regulatory regimes rather than payment screening alone, neutral third-party signing rather than self-signing by the screening party, organization-wide control attestation via controls_evaluated rather than a single admission verdict, and ML-DSA-65 post-quantum signing. New informative references: [A2A], [A2A-IDF], [DRAFT-HOPLEY-X402].

Changes in draft -04

Cross-agent integrity revision. The threat case is a compromised intermediary that swaps payload bytes between two honest agents while both per-agent hash chains validate independently.

* Section 5.3 digest-scope correction. The chain-link digest scope is the canonical signing-input bytes (the JCS-canonical serialization of the predecessor's signed payload object, the same bytes the predecessor's signature covers), NOT the envelope object that additionally includes the signature or anchors top-level keys. The earlier -04 text that said the digest covers "the entire signed receipt object including the signature field" was a Security Considerations over-reach that did not match the reference implementation and would have required every deployed chain to be re-issued. The corrected scope matches both the reference implementation and the upstream [ACTA-RECEIPTS] Section 5.7 rule. The security rationale is updated accordingly: the chain binds the signed-over content of the predecessor (recomputable offline from the predecessor's payload alone), and cross-agent envelope-including-signature integrity is delegated to counterparty_binding (Section 5.6), which IS at the envelope-including-signature scope precisely because the peer signature is the load-bearing artefact in that case. Section 9.1 is updated to match.

* New informative Section 10.2 in Security Considerations acknowledges that the single-linear per-agent chain rule of Section 5.3 creates a per-issuer serialization bottleneck: a denial-of-service against the predecessor pointer (database lock,

network partition, slow IO) bounds chain throughput. Mitigation: issuers SHOULD use bounded predecessor-lookup timeouts, emit a structured protectmcp:lifecycle audit event with a stable chain_emission_blocked reason code when the timeout fires, and document a chain-head recovery procedure for crashed emitters. Parallel per-issuer throughput beyond a single linear chain remains the existing distinct-issuer_id escape hatch.

* Section 10.12 residual list is rewritten for honesty. The M+B collusion residual is downgraded from "M+B alone defeats counterparty_binding" to "M+B+(A's-storage compromise or unavailability) defeats counterparty_binding": the audit-time verifier resolves receipt_ref to A's retained envelope and recomputes the digest, so an honest reachable A's storage defeats M+B collusion alone. A new M+A collusion residual is added: M and A coordinating to produce a fraudulent A-signed envelope is fundamentally outside the receipt model's threat surface because every cryptographic invariant holds; mitigation is separation of duties between issuer and intermediary plus anchor evidence on independent witnesses under different trust roots.

* The worked-example appendix carries an explicit disclaimer that the keys are shown in human-readable order (not JCS-canonical lexicographic order), the sig/value/hash bytes are deterministic placeholders, and implementations MUST NOT replay or trust the example as a real receipt. The disclaimer states the JCS-canonical key ordering that an implementer would derive on the wire.

* Section 5.6.1 envelope_hash rule is tightened on three axes. First, the base64 alphabet is no longer ambiguous: standard base64 per [RFC4648] Section 4 on emission, OR base64url per Section 5 where the transport requires URL-safe encoding, and verifiers MUST accept both and normalise before comparison. Second, JSON-framed digest scope is now normative and mandatory-to-implement: the JCS-canonical UTF-8 byte sequence of A's signed envelope JSON object per [RFC8785], where the envelope is the three-key object {"payload", "signature", "anchors"} with anchors OPTIONAL and B forbidden from re-canonicalizing or stripping any of the three keys before computing the digest. Third, cross-framing equivalence is explicitly addressed: JCS-canonical JSON, COSE deterministic encoding, and JWS Compact Serialization with JCS-canonical payload produce different byte sequences from the same semantic payload-and-signature, so envelope_hash is framing-specific; a transcoding intermediary that re-frames A's envelope MUST be treated as a tampering event, and verifiers MUST NOT reframe before recomputing.

* Section 11.1 entries now carry an explicit Scope tag (signed-payload for risk_class, incident_class, counterparty_binding; envelope-level for anchors). The CWT claim request for counterparty_binding now proposes allocation from the IANA First Come First Served range of [RFC8392] Section 9.1 rather than the unqualified "unassigned integer range" placeholder, so as not to consume Expert Review or Standards Action space.

* The [CIRCIA] reference is corrected to cite the statutory authority (Public Law 117-103 Division Y, codified at 6 U.S.C. 681 et seq.) rather than CISA's general topic page; the March 15, 2022 enactment date is retained, and the pending NPRM at 89 FR 23644 (April 4, 2024) is named in the reference annotation.

* New normative Section 4.6 (Section 5.6) defines the counterparty_binding extension field, an in-payload object carrying a base64-encoded SHA-256 digest (envelope_hash) over the full signed envelope of a peer agent (including the peer's signature bytes), a resolvable opaque receipt_ref, an OPTIONAL expect_ack_from identifier (kid/issuer_id of the expected acknowledger), and an OPTIONAL operational transport_label. Section 4.6.3 (Section 5.6.3) defines the MUST-reject rule when the bound digest does not resolve, the storage obligation on the Audit Pack production layer, and the pairwise default for N-greater-than-2 chains.

* The anchors top-level array schema is now defined inline in Section 5.4: each entry MUST carry type and value (base64-encoded anchor token bytes), with OPTIONAL informational status (anchored / pending / failed) and bitcoin_block (Bitcoin block hash for upgraded OpenTimestamps entries). The IANA Extension Fields Registry entry for anchors is updated to reflect the four-member schema.

* The IANA Type Namespaces Registry initial contents now include the registered sub-namespace protectmcp:lifecycle:configuration_change emitted by the reference cloud implementation for configuration-change receipts under Section 6.1.1.

* CIRCIA retention floor (Section 7.7.2) is rewritten to measure two years from the submission of the most recently required CIRCIA report per proposed Section 226.13(c) of the CIRCIA NPRM at 89 FR 23644 (April 4, 2024), not from the date of the underlying Action.

* HIPAA retention (Section 7.4.2) is rewritten to cite 45 CFR 164.316(b)(2) with the six-year floor expressed as "six years from the date of creation or the date when last in effect, whichever is later" and the analogy basis for applying that floor to audit-log content explicitly named.

* EU AI Act Article 26(6) retention (Section 6.1.5) is rewritten to express the six-month floor in calendar-arithmetic terms per [ISO8601-2]; the 184-day day-count figure is retained as an informative anchoring-interval floor, and the previously-endorsed 183-day pick is withdrawn.

* Section 5.1.3 now states explicitly that issuer_id values MUST be bare identifiers without a scheme prefix where the scheme is unambiguous (LEI: 20-character alphanumeric self-identifying form), aligning the spec with the reference cloud emitter under the cloud-wire-conformance change.

* New informative Section 9.11 (Section 10.12) documents the residual threat that counterparty_binding partially mitigates (M silently swaps bytes between honest A and B) and names three operational mitigations Operators SHOULD adopt: multiple independent anchor witnesses, regulator-driven side-by-side chain comparison under SEC 17a-4 audit-trail workflow, and out-of-band M relay logs.

* Section 7 (Section 8) records two manifest-level fields shipped in the reference Audit Pack producer: regime_mapping_disclaimer (when per-receipt regime predicates derive from a producer-side mapping the original signer did not co-sign) and stale_pending (per-entry flag set when anchor evidence remains pending after the bound of Section 5.4).

* Section 8.3 (Section 9.3) replaces the prior one-paragraph clause with five SHOULD-emit per-axis fields: regimes_satisfied, anchor_valid_ots, anchor_valid_rfc3161, policy_digest_resolved, duplicate_emission_candidate.

* New informative Section 9.9 (Section 10.10) enumerates the residuals counterparty_binding does not solve. New informative Section 9.10 (Section 10.11) records channel-level operator guidance citing [RFC8446], [RFC9266], [RFC5705], and [RFC9421] and disclaims transport-layer security as a substitute for application-layer byte equality.

* Section 11 (Section 11) adds counterparty_binding to the Extension Fields Registry and requests registration as a CWT claim per [RFC8392].

* New normative references: [RFC9052], [RFC8949], [RFC7515], [RFC8392]. New informative references for channel-level guidance only: [RFC8446], [RFC5705], [RFC9266], [RFC9421].

* New normative Section 4 bounds the inputs to which the inherited JCS rule of [RFC8785] is applied: IEEE-754 floating-point numbers MUST NOT appear in the digest-covered canonical form (callers SHOULD serialize numerics as JSON strings or as integer-rational pairs in the IEEE-754 safe range), and tool-version-specific semantic equivalence (SQL case folding, path normalization, locale-aware string collation, numeric tolerance, URL percent-encoding choices) is OUT OF SCOPE for the chain layer. The chain answers byte equality for one agent identity at one wall-clock time only; semantic equivalence belongs in the policy_digest artefact and the Audit Pack manifest.

* Section 5.3 now states explicitly that each issuer MUST maintain a single linear per-agent chain and MUST serialize concurrent in-agent emission (parallel tool calls, thread-pool fan-out) through a single predecessor pointer in emission order. Parallel sub-chains within one agent identity (a per-receipt chain_id discriminator) are NOT defined by this profile; an issuer that requires parallel sub-chains MUST express each parallel path as a distinct agent identity with its own issuer_id, signing key, and chain rooted at the all-zero genesis value.

* Wire tightenings introduced in this revision (additive at the message level, restrictive at the verifier level): (a) the anchors entry value member is now REQUIRED where the upstream profile leaves it OPTIONAL; (b) issuer chains are normatively single-linear per issuer (no parallel chain_id discriminator), so a -03 producer that ran parallel sub-chains under one issuer_id emits non-conformant -04 receipts; and (c) IEEE-754 floating-point numbers MUST NOT appear in the canonical form covered by SHA-256 digests. A -03 receipt with a missing anchor value, parallel sub-chains, or a digest-covered float is not a conformant -04 receipt. Implementations targeting -04 SHOULD re-emit -03 receipts under -04 emission rules. The signing algorithms, the canonicalization transformation itself (JCS), the anchor types, and the regime bindings of Sections 5 and 6 are unchanged. A -03 verifier remains conformant for receipts that do not carry counterparty_binding; -03 verifiers encountering the field will ignore it per [ACTA-RECEIPTS] Section 4.2 extension semantics.

* Rationale paragraph of Section 4 is rewritten to ground the IEEE-754 float ban on documented divergence between mainstream JSON serializers (Python json.dumps, Go encoding/json, Java Jackson) and the ECMA-262 Number-to-String algorithm referenced by

Section 3.2.2.3 of [RFC8785], rather than on a misstated claim that JCS itself fails to specify float serialization outside the safe integer range. The Unicode-normalization bullet of Section 4 is corrected to acknowledge that Section 3.1 of [RFC8785] mandates as-is preservation of Unicode strings (no NFC default).

* Security Considerations Section 10.1 is corrected to reflect the tiered DORA deadlines: the four-hour initial-notice clock (with 72-hour intermediate-report and one-month final-report bounds) per DORA Article 17 and the RTS in [REG-2025-301], rather than a flat 72-hour reporting deadline. New informative reference [REG-2025-301] added.

* The Annex II field 3.23 (Type of the incident) citation in Section 5.5 and the incident_class initial-registry entry of Section 11.1 no longer reproduces the enumeration verbatim; verifiers MUST resolve the canonical values from [REG-2025-302] directly.

* Section 5.2 extends the wire decision vocabulary with observation, a fourth value reserved to type protectmcp:lifecycle for receipts emitted when an Action was signed without any policy evaluation. The new value is the regulator-honest alternative to a misleading allow on the "no policy matched" path; emitters MUST refuse to issue a protectmcp:decision receipt that carries observation, and verifiers MUST reject the combination. The vocabulary-namespace registry entry for protectmcp:decision in Section 11.2 is updated to reflect that observation is reserved to protectmcp:lifecycle.

* New informative appendix (Appendix "Appendix - Capture Topologies for Compliance Receipt Emission") catalogues six capture topologies an operator can use to emit conformant Compliance Receipts in environments where the originating application code cannot be modified to call the receipt-emitting SDK directly: in_process_sdk, network_proxy, browser_extension, ebpf_observer, mcp_proxy, passive_telemetry. The appendix is non-normative; the capture_topology attribute it defines is OPTIONAL at the wire layer and, where present, lives in the Audit Pack manifest entry rather than inside the signed payload object, so the topology declaration does not alter signed bytes. The six values form a closed initial vocabulary at this revision; Appendix "capture_topology Vocabulary and Considerations for a Future IANA Registry" sketches a future "Compliance Receipt Capture Topologies" IANA registry under the "Compliance Receipts" registry group with Specification Required registration policy per [RFC8126] for a follow-on revision, and names reserved-value avoidance guidance for early extenders.

Changes in draft -03

Multi-jurisdiction consolidation. The European Union profile (formerly the only profile in -02) and the United States profile (formerly the separate draft draft-marques-asqav-us-compliance-receipts-00) are merged into a single document with two regional bindings sections: Section 5 (European Union) and Section 6 (United States). Sections 1 through 4 (Introduction, Conventions, Relationship to upstream, Receipt Field Profile) and Sections 7 through 11 (Audit Pack, Verifier, Security, IANA, Acknowledgements) are shared across both regimes. Conventions terms that differ across regimes are now disambiguated with regime suffixes (Deployer (EU AI Act) vs Deployer (Colorado AI Act); High-Risk AI System (EU AI Act) vs High-Risk AI System (Colorado AI Act)). The incident_class extension field now lists every applicable canonical category in one place: ICT-related incident under [DORA] with the Annex II reporting enumeration, Cybersecurity Event/Incident under [NYDFS-500], Covered Cyber Incident under [CIRCIA], and security incident under [HIPAA-SECURITY]. The issuer_id rule now permits EIN or CIK as alternatives to LEI for US Deployers without an allocated LEI. The Tamper Resistance security consideration extends the one-hour anchoring SHOULD to NYDFS 500.17 and CIRCIA in addition to DORA Article 17. The Privacy security consideration extends to GDPR for EU data subjects and CCPA / VCDPA / HIPAA Privacy Rule for US data subjects. The Worked Example notes that the wire shape applies identically to US bindings, with only the issuer_id identifier and the vocabularies differing. IANA registries are unchanged; the Initial registry contents for incident_class now describe the multi-regime category set. No changes to the wire format, the field profile, the hash chain, the anchoring rules, the Audit Pack contents, or the Verifier checks. Section 4.1.6 (sandbox_state) and Section 4.5 (risk_class) corrected to attribute the EU risk-management documentation requirement to Article 9 of [EU-AI-ACT] (Provider’s risk management system) rather than Article 26, with Article 26(1) cited as the deployer’s instructions-for-use obligation that links to the Provider’s Article 9 documentation. Section 6.5.3 (nydfs-retention) corrected to a single-tier five-year floor under 23 NYCRR 500.6(b) (verified verbatim against LII Cornell); the prior tiered 5-year/3-year split (claimed against the DFS Second Amendment) was incorrect because the Second Amendment does not amend Section 500.6, leaving the 2017 single-tier text in force. The 1096-day three-year audit-trail floor previously stated for NYDFS is removed.

Changes in draft -02

Submission-ready EU-only profile. Wire-shape alignment with upstream [ACTA-RECEIPTS] (payload/signature/anchors envelope; payload_digest object form; tool_name REQUIRED for protectmcp:decision; issuer_id equals kid). EU AI Act and DORA bindings authored against Official Journal text. Anchor MUST (at least one of RFC 3161 or OpenTimestamps); both RECOMMENDED; 7-day OpenTimestamps upgrade deadline profile-imposed. Six-month AI Act floor expressed as 184 days; DORA-bound default expressed as 1827 days. IANA registries created.

Changes in draft -01

Initial wire-shape alignment with upstream and addition of dual-anchor, hash-chain, retention, and DORA classification bindings. Subsequent revisions superseded the specific values introduced here.

Changes in draft -00

Initial version. Defines a profile of [ACTA-RECEIPTS] that binds receipt fields to EU AI Act Article 12, EU AI Act Article 26, and DORA Article 17.

Appendix - Capture Topologies for Compliance Receipt Emission

This appendix is informational and non-normative. It catalogues six capture topologies an operator can use to emit conformant Compliance Receipts in environments where the originating application code cannot be modified to call the receipt-emitting SDK directly. The topologies are listed in order of payload fidelity, from highest (in-process SDK, full payload digest) to lowest (passive telemetry, post-hoc log ingestion only). All six emit receipts that satisfy the wire profile of Sections 3 and 4; they differ in trust boundary, payload coverage, and the operational identity that the receipt binds. The intent is to give an operator vocabulary for declaring which topology produced a given receipt, so that a verifier or auditor can interpret the receipt’s evidentiary weight without re-deriving the architecture from out-of-band documentation.

An operator MAY declare the producing topology by including a capture_topology attribute in the Audit Pack manifest entry for the receipt. The attribute is informational at the wire layer and OPTIONAL. The vocabulary defined below is closed for the topologies catalogued in this appendix; future revisions or third-party profiles MAY extend it through the future-IANA-registry mechanism noted at the end of this appendix.

In-Process SDK

The originating application links the receipt-emitting SDK directly and calls it inline with the action being recorded. This is the baseline pattern [ACTA-RECEIPTS] and Sections 3 and 4 of this document are written against. The receipt carries a full payload_digest covering the action's request bytes; the SDK has direct access to the application's principal identity, the policy decision, and the request body. Vocabulary value: in_process_sdk. Trust boundary: the application process itself; the SDK runs inside the application's memory space and inherits its principal. Threat-model note: captures the full request and response payloads, the deciding principal, and the policy context; does NOT capture out-of-process side effects or actions taken by sibling processes that do not link the SDK. Reference implementation hint: the Asqav Python and TypeScript SDKs published under the asqav-sdk umbrella (Apache-2.0) implement this pattern; an operator MAY substitute any other conformant [ACTA-RECEIPTS] implementation.

Network-Layer Egress Proxy

A customer-owned reverse proxy or egress gateway (Envoy, NGINX, or an equivalent forward proxy) sits in the network path between the application and the downstream LLM provider. The proxy tees the request to a co-located Compliance Signer process, which receives the request bytes, applies the policy evaluation, and emits a receipt via a signer RPC. A DNS-rewrite on-ramp (CoreDNS rewriting the LLM hostname to the proxy address) or a Server Name Indication (SNI) router (SNIProxy at Layer 4) MAY be used to force application traffic onto the proxy without per-application configuration. Vocabulary value: network_proxy. Trust boundary: the customer's network egress; the proxy and signer run under the customer's operational control, and the receipt is signed by a key the customer's signer holds. Threat-model note: captures the request and response bytes that traverse the proxy and the network-layer principal identity (source IP, mTLS client cert if present); does NOT capture traffic that bypasses the proxy (direct outbound from a non-routed host, DNS-over-HTTPS to a hard-coded resolver, or TLS connections to certificate-pinned endpoints that the customer's enterprise CA cannot inspect). The receipt's issuer_id binds the customer's signer, not the originating application; the application's identity, where captured, appears as an attribute resolved through the Audit Pack manifest. Reference implementation hint: CoreDNS (Apache-2.0) for the DNS on-ramp, SNIProxy by dlundquist (BSD-2-Clause) for the SNI router, and Envoy (Apache-2.0) for the Layer-7 reverse proxy plane; none of these are normative requirements.

Browser Extension

A managed-browser Manifest V3 (MV3) extension installed on employee workstations via the enterprise Mobile Device Management (MDM) system intercepts fetch, XMLHttpRequest, and EventSource requests to a configured list of LLM hostnames. The extension POSTs the intercepted request and response bytes to a Compliance Signer endpoint, which emits the receipt. For LLM hosts that pin their TLS certificates, the customer's enterprise root Certificate Authority (CA) MUST be installed in the browser trust store via MDM so that the extension's content-script interception can observe decrypted bytes. Vocabulary value: browser_extension. Trust boundary: the managed browser process on the employee workstation; the extension runs under the browser's sandbox and the employee's interactive session. Threat-model note: captures the full request and response payload for LLM calls initiated from the browser by the human user, and binds the receipt to the browser's principal identity (the user's enterprise single-sign-on subject, where the extension can read it). Does NOT capture LLM calls made by native desktop applications, server-side daemons, or browsers without the extension installed; does NOT capture traffic in incognito or private-window modes unless the extension is explicitly authorised for those contexts. Reference implementation hint: the open-source Chrome MV3 extension scaffolding published by Google under the chrome-extensions-samples repository (Apache-2.0) is a useful starting point; the customer's signer endpoint is the Asqav signer or any conformant [ACTA-RECEIPTS] implementation.

eBPF SNI Observer

A kernel-level extended Berkeley Packet Filter (eBPF) probe attached to the host's network stack observes outbound TLS ClientHello records. The probe extracts the SNI hostname, the JA3 client fingerprint, the source and destination addresses and ports, and the connection timestamp, and emits a lower-fidelity receipt that binds the employee, the device, the wall-clock time, and the LLM host without observing payload content. Vocabulary value: ebpf_observer. The receipt SHOULD carry an informational payload_capture attribute set to false in the Audit Pack manifest, so that a verifier or auditor does not assume payload-level evidence is recoverable. Trust boundary: the kernel of the host on which the probe runs; the probe operates below the application's user-space process and observes traffic regardless of application configuration. Threat-model note: captures the existence and counterparty of an LLM call (the "did the call happen" evidence class) and the device-and-employee binding through host attestation; does NOT capture request or response bytes, the prompt content, the model parameters, or the decision-relevant context. Useful when payload capture is operationally infeasible

(TLS certificate pinning that the enterprise CA cannot defeat, third-party SaaS that egresses outside the customer's proxy plane) but the operator still needs evidence that a regulated LLM interaction occurred. Reference implementation hint: Inspektor Gadget (Apache-2.0) and Tetragon (Apache-2.0) both expose SNI and connection-metadata events from eBPF probes; neither is a normative requirement.

MCP Transparent Proxy

A transparent proxy sits in-path between a Model Context Protocol (MCP) client and one or more downstream MCP servers, terminating the client's stdio, Server-Sent Events (SSE), or streamable-HTTP transport on one side and re-establishing the same transport to each downstream server on the other side. The proxy observes every tools/call and resources/read JSON-RPC method invocation, signs a receipt at the moment the call is forwarded, and emits a second acknowledging receipt carrying a counterparty_binding (see Section 5.6) at the moment the downstream server's response returns. Both sides of the call are therefore bound bilaterally, with the proxy's signing key serving as the integrity anchor for the pair. Vocabulary value: mcp_proxy. The receipt SHOULD carry the invoked MCP method (for example, tools/call or resources/read) as an attribute in the Audit Pack manifest entry, so that a verifier can filter by call type without re-parsing payload bytes. Trust boundary: the proxy process and its signing key; the upstream MCP client and the downstream MCP server are both treated as honest endpoints under the threat model of Section 10.12, with the proxy itself being the named intermediary whose tampering risk counterparty_binding mitigates. Threat-model note: captures the full MCP request and response payload, the method name, and the client and server principal identities visible at the transport boundary; does NOT capture MCP traffic that bypasses the proxy or that uses a transport the proxy does not implement. Reference implementation hint: the Asqav MCP transparent proxy published under the asqav-mcp repository (Apache-2.0) and the upstream FastMCP project (MIT) on which it builds; neither is a normative requirement.

Passive Telemetry Ingestion

A passive ingestion pipeline reads structured records the originating application or its runtime has already emitted (for example, OpenTelemetry spans, application access logs, vendor-managed observability exports, or batch CSV drops) and synthesises a Compliance Receipt for each record after the fact. The synthesiser holds the signing key, applies the receipt-format wire profile, and emits the receipt to the same downstream sink that the in-process SDK and network-proxy paths feed. Vocabulary value: passive_telemetry. The receipt SHOULD carry an informational payload_capture attribute

set to the value the underlying telemetry source actually preserved (full body, headers only, or none), so that a verifier or auditor can interpret the receipt's evidentiary weight against the upstream pipeline's retention policy. Trust boundary: the telemetry pipeline operator's signing key plus the integrity of the upstream observability source; the receipt binds the producer of the telemetry, not the originating application's per-request principal. Threat-model note: captures whatever the upstream telemetry source preserved (typically a subset of the action's bytes and metadata, often without request or response payload) plus the wall-clock and counterparty identifiers visible in the telemetry record; does NOT capture data the upstream source dropped, sampled out, or never emitted, and inherits any tampering risk the upstream source carries between emission and ingestion. Useful when the in-process SDK, network proxy, browser extension, eBPF observer, and MCP proxy topologies are all operationally infeasible (legacy applications without instrumentation hooks, third-party SaaS with read-only export, fleet migrations where the producer has only logs to work from) but the operator still needs a signed evidence artefact tied to the historical action. Reference implementation hint: any OpenTelemetry collector exporter (Apache-2.0) feeding a conformant receipt-emitting signer; the upstream telemetry source is out of scope of this profile.

capture_topology Vocabulary and Considerations for a Future IANA Registry

The six values defined in this appendix (in_process_sdk, network_proxy, browser_extension, ebpf_observer, mcp_proxy, passive_telemetry) form the closed initial vocabulary for the capture_topology attribute. The attribute is OPTIONAL at the wire layer and, where present, MUST appear in the Audit Pack manifest entry for the receipt rather than inside the signed payload object, so that the topology declaration is producer-side metadata that does not alter the receipt's signed bytes. A verifier MUST NOT treat the absence of a capture_topology attribute as a non-conformance condition; absence simply means the producer did not declare a topology.

This document is an Independent Submission and does not request IANA action for the capture_topology vocabulary at this revision. A future revision MAY request creation of a "Compliance Receipt Capture Topologies" registry under the same "Compliance Receipts" registry group described in Section 11, with the registration policy of Specification Required per [RFC8126] and an initial set populated from the six values above. Until such a registry exists, implementations that extend the vocabulary SHOULD document the new value in a reference specification and SHOULD avoid colliding with

the six reserved values above. The Designated Expert(s) for any future registry SHOULD verify that a candidate value names a distinct topology (a different trust boundary or a materially different payload-fidelity class) rather than a variant of an existing one, and that the candidate's threat-model note states what is captured and what is not, in the form used by the six entries in this appendix.

Author's Address

Joao Andre Gomes Marques
Asqav
Portugal
Email: joaoagm90@gmail.com