Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2026

Y. B. Lee
Meridian Verity Group

4 June 2026

Permit Receipts for Permit-Before-Commit Authorization of AI-Agent and Workload External Effects
draft-lee-orprg-permit-receipts-00

Abstract

This document defines requirements and an abstract data model for PermitReceipts used in permit-before-commit authorization of AI-agent and workload external effects. A verifier evaluates a canonicalized effect request, action digest, policy epoch, validity interval, scope, issuer evidence, revocation status, and anti-replay state before a protected effect is committed at an effect boundary. The document specifies verifier behavior, failure semantics, conformance expectations, and candidate interoperability registries for discussion. It is intended to enable IETF discussion about the appropriate home, scope, and wire-profile split for this work.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 December 2026.

Copyright Notice

Table of Contents

1. Introduction

AI agents and automated workloads increasingly initiate external effects, including tool calls, retrieval operations, data egress, transactions, configuration changes, extension updates, inter-agent messages, key-release operations, and output releases. Controls evaluated only at a session, prompt, or application-entry boundary can become stale or bypassable before a concrete external effect is committed.

This document treats protected external effects as commit events. A boundary interceptor captures the proposed effect before commitment, canonicalizes the effect request, computes an action digest, and invokes a verifier. The verifier returns ALLOW only when a

PermitReceipt and the relevant policy, epoch, recency, scope, issuer, and anti-replay evidence satisfy the selected verification profile. Otherwise, the verifier returns DENY with a structured denial reason.

The design intentionally makes a narrow, testable security claim: a protected external effect is not committed unless machine-verifiable evidence authorizes the exact effect under current policy state. The design does not require the AI model or workload to be honest, aligned, or able to explain its intent; it treats the model or workload as a request producer whose effect-bearing requests require independent authorization at the effect boundary.

This -00 individual draft is offered for security-area discussion and dispatch guidance. It asserts no IETF consensus. It defines terminology, requirements, an abstract receipt model, verifier behavior, failure semantics, operational considerations, and candidate registries. It does not define a complete policy language, does not select a mandatory wire format, and does not claim production non-bypassability for any deployment by itself.

2. Terminology and Requirements Language

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.

External effect
: An operation that crosses an execution boundary and may disclose information, change state, invoke a privileged operation, release output, delegate authority, or alter future effect semantics.

Effect boundary
: A logical or physical boundary between an execution substrate and an external interface at which an external effect would be committed.

PermitReceipt
: A machine-verifiable authorization artifact that binds an external-effect request to policy, epoch, validity, scope, action binding, authenticity evidence, and required status evidence.

Canonical request representation
: A deterministic representation of an external-effect request containing the effect-relevant fields for a selected canonicalization profile.

Action digest
: A digest computed over the canonical request representation.

Verifier
: A component that evaluates a request, PermitReceipt, policy state, revocation state, and explicit context, and returns ALLOW or DENY with evidence digests and a denial reason when applicable.

Fail closed
: The default behavior in which missing, stale, ambiguous, conflicting, malformed, revoked, replayed, unsupported, or unverifiable evidence results in DENY.

3. Problem Statement

Let S be an execution substrate, I be a protected external interface, e be an effect request, C(e) be the canonical request representation, H(C(e)) be the action digest, Pt be policy state, Rt be revocation state, ctx be explicit context, and r be a PermitReceipt. The objective is:

Commit_I(e) occurs only when Verify(e, r, Pt, Rt, ctx) = ALLOW and all policy-designated downstream capability checks succeed. Otherwise the result is DENY.

An adversary can influence prompts, retrieved content, tool arguments, scheduled workflows, plugin state, non-canonical encodings, local application paths, and background triggers. The adversary can try to cause unauthorized effects, reuse stale authorization, substitute materially different actions after authorization, induce ambiguous context selection, replay receipts or capabilities, suppress revocation evidence, or reach an external interface without the intended authorization path.

This document addresses the authorization predicate at the effect boundary. Policy correctness, root-of-trust integrity, model alignment, and complete production placement are necessary deployment concerns but are not solved solely by this protocol profile.

4. Core Requirements

REQ-1 A protected external effect MUST be evaluated before commitment at the applicable effect boundary.

REQ-2 The verifier MUST bind authorization to a canonical request representation and an action digest or equivalent cryptographic commitment.

REQ-3 The verifier MUST bind authorization to a policy digest or policy epoch, and MUST deny on epoch mismatch or rollback when policy requires strict or minimum-epoch compatibility.

REQ-4 The verifier MUST evaluate time-bounded validity and anti-replay material when required by the selected verification profile.

REQ-5 The verifier MUST evaluate revocation or status evidence to the recency required by policy for the receipt, issuer, authority profile, policy epoch, and other required evidence.

REQ-6 The verifier MUST deny when authorization-critical context is missing, ambiguous, conflicting, stale, revoked, replayed, malformed, unsupported, or unverifiable, unless a narrowly scoped constrained mode is explicitly authorized by policy.

REQ-7 Deployments claiming non-bypassability MUST ensure that protected effect-bearing paths traverse an interceptor or are rejected by a downstream verifier requiring valid ORPRG evidence.

REQ-8 Implementations SHOULD emit auditable decision records containing action digest, policy digest, epoch, outcome, denial reason if any, and evidence references sufficient for independent reproduction when the referenced evidence is available.

5. Architecture

An ORPRG deployment consists of an effect-boundary interceptor, canonicalizer, action-digest engine, PermitReceipt handler, verifier, policy and epoch source, revocation or status source, optional transparency log, audit logger, and optional downstream capability verifier.

The interceptor captures a protected external-effect request before commitment. The canonicalizer produces a deterministic canonical request representation that includes all effect-relevant fields for the selected effect type and canonicalization profile. The digest engine computes the action digest. The verifier evaluates the PermitReceipt and associated evidence before the effect crosses the boundary.

A protected external interface can include an egress gateway, retrieval gateway, service-mesh sidecar, API gateway, message broker, privileged API boundary, scheduler boundary, update-channel gate, key-release gate, kernel or hypervisor hook, output-release boundary, or recipient-side verifier.

Dual enforcement is RECOMMENDED for high-risk effects. Under dual enforcement, successful local verification produces a DecisionReceipt or capability token bound to the action digest, audience, policy digest, epoch identifier, validity interval, and anti-replay material. The downstream interface denies commitment if this derived evidence is absent, expired, replayed, audience-invalid, mismatched, or unverifiable.

6. Protected Effect Types

ORPRG is intended for effects whose commitment can change external state, disclose information, release output, delegate authority, or change future effect semantics. Example effect families include:

* network transmission and data egress;
* data access and retrieval, including database queries, file reads, object-store access, and retrieval-augmented generation fetches;
* tool calls, privileged API calls, transactions, and scheduler operations;
* key-release, signing, decrypt, unwrap, or credential-broker operations;
* inter-agent messages and protocol or representation negotiation;
* plugin, extension, policy, model-artifact, or configuration updates; and
* output release to a user or downstream system when policy designates the release as protected.

Each effect family requires a profile that identifies the effect-relevant fields to be canonicalized and included in the authorization decision. A receipt issued for one effect type, target, tenant, purpose, jurisdiction, audience, key operation, or budget MUST NOT authorize a materially different effect.

7. PermitReceipt Abstract Data Model

A PermitReceipt is a structured authorization artifact. This document defines an abstract model and example JSON shape. Future documents or revisions can define concrete serializations, such as JSON with a specified canonicalization profile, CBOR/COSE, or JOSE. JSON profiles can reuse or adapt JSON Canonicalization Scheme (JCS) [RFC8785] when its assumptions fit the selected effect type.

```
{
  "receipt_core": {
    "policy_digest": "hex-or-base64url",
    "epoch_id": "string-or-integer",
    "valid_from": "time-or-sequence",
    "valid_to": "time-or-sequence",
    "action_digest": "hex-or-base64url",
    "action_commitment": "optional commitment object",
    "canonicalization_profile": "profile identifier or digest",
    "scope": {
      "effect_type": "effect type name",
      "interface_id": "string",
      "target_id": "string",
      "tenant_id": "optional string or digest",
      "purpose_id": "optional string",
      "jurisdiction": "optional string or set",
      "audience": "optional downstream verifier identifier",
      "budget": "optional structured limit"
    },
    "anti_replay": {
      "nonce": "optional",
      "sequence_number": "optional",
      "monotonic_counter": "optional"
    }
  },
  "verification_profile": {
    "authority_profile_id": "optional",
    "assurance_level_id": "optional",
    "permit_class_id": "optional"
  },
  "permit_provenance": {
    "permit_artifact_ref": "optional",
    "permit_provenance_digest": "optional"
  },
  "authenticity": {
    "issuer_id": "string",
    "signature": "signature value",
    "issuer_chain": "optional",
    "threshold_signature": "optional"
  },
  "status": {
    "revocation_status_proof": "optional",
    "signed_checkpoint": "optional",
    "inclusion_or_non_inclusion_proof": "optional"
  }
}
```

The PermitReceipt MUST bind authorization to either an action digest, a cryptographic commitment to the canonical request representation, or both. If both are present, implementations MUST verify both. A receipt MUST be evaluated under the canonicalization profile and policy epoch for which it was issued or under a policy-defined compatibility rule.

When the selected verification profile requires permit provenance, identity binding, purpose binding, jurisdiction binding, assurance evidence, or attestation evidence such as RATS-style evidence [RFC9334], the verifier MUST treat missing or unverifiable evidence as DENY.

8. Verifier Result

A verifier returns a structured result. At minimum, the result contains a decision. For DENY, it SHOULD contain a denial reason. For auditability, it SHOULD contain evidence digests and recency observations. Example shape:

```
{
  "decision": "ALLOW | DENY",
  "denial_reason_code": "optional string",
  "evidence_digests": {
    "action_digest": "hex-or-base64url",
    "receipt_digest": "optional",
    "policy_digest": "hex-or-base64url",
    "epoch_id": "string-or-integer",
    "status_evidence_digest": "optional",
    "checkpoint_digest": "optional"
  },
  "recency_observations": {
    "status_age_seconds": "optional",
    "checkpoint_age_seconds": "optional"
  },
  "constrained_mode": "optional policy-defined status"
}
```

A verifier result MUST NOT be interpreted as authorization for a different action digest, audience, policy epoch, scope, or validity interval. A downstream verifier that consumes a DecisionReceipt or capability token MUST validate the audience, expiry, anti-replay material, and action binding before allowing commitment. Deployments MAY use proof-of-possession or token-exchange patterns such as [RFC9449] and [RFC8693] where appropriate, but such tokens MUST remain bound to the verifier outcome required by policy.

9. Canonicalization and Action Digests

Canonicalization is a security boundary. A canonicalization profile MUST define how effect-relevant fields are selected, normalized, ordered, encoded, and rejected. Profiles need to address duplicate fields, default values, Unicode normalization, numeric representation, array ordering, header and query-parameter normalization where applicable, and effect-specific semantics.

The action digest MUST be computed over the canonical request representation using the digest algorithm required by the selected profile. A receipt MUST identify or be bound to the canonicalization profile used for issuance. If the verifier cannot establish canonicalization-profile compatibility, it MUST return DENY.

Profiles SHOULD include equivalent-input and distinct-effect test vectors. Equivalent inputs are representations that should produce the same action digest. Distinct effects are requests that are materially different and should produce distinct action digests.

10. Verifier Behavior

The verifier evaluates a canonical request, PermitReceipt, policy state, status evidence, and explicit context. The verifier behavior is a total function: every recognized failure maps to DENY with a reason; every malformed, unknown, or unsupported state maps to DENY rather than to ALLOW.

1. Validate request and receipt schemas before semantic authorization.

2. Compute or validate the canonical request representation and action digest.

3. Verify authenticity evidence and issuer or authority-profile trust.

4. Verify that the receipt’s action digest or commitment authorizes the exact request.

5. Verify time-bounded validity, clock or sequence requirements, and anti-replay material.

6. Verify policy epoch compatibility and minimum-epoch requirements.

7. Verify scope, identity, purpose, jurisdiction, audience, budget, key-operation, and provenance constraints when required.

8. Verify status and revocation evidence to the recency required by the selected profile.

9. Emit a verifier result and audit evidence.

10. Return ALLOW only if all required checks succeed; otherwise return DENY.

11. Policy Epochs, Revocation, and Recency

A PermitReceipt is not authorized merely because its signature verifies. It also needs to be compatible with the current policy epoch, within its validity interval, and supported by sufficiently fresh status evidence when such evidence is required by policy.

Status evidence can include signed revocation lists, signed revocation event records, issuer-credential status, authority-profile status, policy-epoch status, signed checkpoints, inclusion proofs, non-inclusion proofs, or other profile-defined evidence. Profile designs can reuse certificate revocation lists [RFC5280], OCSP-like status [RFC6960], and transparency-log patterns such as Certificate Transparency [RFC6962] where appropriate. If policy requires status recency and the verifier cannot establish that recency, the verifier MUST return DENY.

Policy epochs can be strict, minimum-epoch, compatibility-window, composite, or profile-defined. Epoch mismatch, rollback evidence, split-brain epoch state, or incompatible policy digest MUST result in DENY unless a policy-defined constrained mode explicitly applies.

Intermittently connected deployments MAY define constrained local operation. Such operation MUST be narrow, short-lived, anti-replay protected, auditable, and explicitly authorized by policy. Absence of required status evidence MUST NOT silently become a fail-open authorization for high-risk effects.

12. Conformance Expectations

Conformance for an ORPRG verifier is behavioral. A conforming implementation preserves permit-before-commit, action binding, epoch compatibility, status recency, scope and provenance enforcement, ambiguity-as-denial, anti-replay, downstream-evidence validation when configured, and audit sufficiency.

A conformance corpus SHOULD include positive and negative vectors for at least: missing receipt, invalid signature, untrusted issuer, malformed request, malformed receipt, action-digest mismatch, canonicalization-profile mismatch, expired validity, epoch mismatch,

epoch rollback, stale status, unknown status, confirmed revocation, scope violation, identity mismatch, purpose mismatch, jurisdiction ambiguity, missing provenance, missing assurance evidence when required, capability-token absence, audience mismatch, and replay.

Conformance vectors SHOULD be signed or distributed with a signed manifest. Implementations SHOULD publish a reproducible "run-the-verifier" procedure that maps each vector to the expected ALLOW or DENY decision and denial reason.

13. Candidate Interoperability Registries for Discussion

This section is non-normative and is included to focus community discussion. If the work progresses, future versions can request IANA registries with precise templates, designated-expert guidance, and registration policies following [RFC8126].

13.1. Candidate Denial Reason Codes

Candidate denial reason names include:

* SIGNATURE_INVALID
* ISSUER_UNTRUSTED
* EPOCH_MISMATCH
* VALIDITY_WINDOW_EXPIRED
* SCOPE_VIOLATION
* ANTI_REPLAY_FAILURE
* REVOKED_CONFIRMED
* REVOCATION_UNKNOWN_OR_STALE
* CANONICALIZATION_MISMATCH
* TRANSPARENCY_PROOF_INVALID
* PERMIT_PROVENANCE_INVALID_OR_MISSING
* CAPABILITY_TOKEN_INVALID_OR_MISSING
* AUTHORITY_PROFILE_SELECTION_FAILED
* AMBIGUOUS_CONTEXT

13.2. Candidate Receipt Claim Names

Candidate claim names include policy_digest, epoch_id, valid_from, valid_to, action_digest, action_commitment, canonicalization_profile, scope, anti_replay, authority_profile_id, assurance_level_id, permit_class_id, permit_provenance_digest, issuer_id, signature, issuer_chain, revocation_status_proof, signed_checkpoint, and inclusion_or_non_inclusion_proof.

13.3. Candidate Profile Registries

Future profile registries could include canonicalization profile identifiers, assurance level profile identifiers, effect type identifiers, and verifier result claim names. Such registries should not be requested until the community agrees on the protocol split and at least one concrete wire profile.

14. IANA Considerations

This document makes no IANA requests.

Future versions may request registries for denial reason codes, receipt claim names, canonicalization profiles, effect types, or verifier result claim names. If such registries are requested, the document will provide clear registration templates, registration policies, change rules, and designated expert guidance following [RFC8126].

15. Security Considerations

ORPRG is an authorization mechanism and is security critical. Its security depends on correct boundary placement, deterministic canonicalization, trustworthy issuer keys, accurate policy and authority-profile selection, fresh status evidence, replay protection, and downstream enforcement where configured.

Implementations MUST treat parser failures, unsupported proof types, missing required fields, duplicate-field ambiguity, stale caches, clock uncertainty, conflicting policy epoch sources, status uncertainty, and unsupported canonicalization profiles as DENY unless a constrained mode is explicitly authorized by policy.

Production deployments MUST ensure that alternate network, storage, key-release, scheduler, plugin, update, output-release, or privileged API paths cannot commit protected effects without ORPRG evidence. A verifier library alone cannot prove deployment non-bypassability.

Canonicalization profiles require careful review and differential testing across implementations. Weak profiles can permit action substitution or authorization confusion. Profiles SHOULD be versioned, digest-bound, and included in policy epoch state.

Replay protection for single-use receipts or capability tokens requires durable state appropriate to the deployment. Distributed deployments need tenant-scoped and audience-scoped replay domains and crash-consistent storage.

16. Privacy Considerations

PermitReceipts and audit records can reveal sensitive operational metadata, including targets, tenants, purposes, jurisdictions, key identifiers, route information, capability audiences, and denial reasons. Implementations SHOULD minimize disclosed fields, use digest references where possible, protect logs, define retention periods, and consider selective disclosure for high-privacy deployments.

Transparency logs and conformance artifacts can improve auditability but can also disclose sensitive patterns. Deployments SHOULD separate public conformance registries from private operational status logs unless policy explicitly permits publication.

17. Operational Considerations

Operators need lifecycle processes for policy epochs, authority profiles, canonicalization profiles, issuer keys, status feeds, audit retention, conformance vectors, incident response, and constrained offline modes. These are not optional deployment details; they are inputs to the verifier’s authorization decision.

High-assurance deployments should identify all effect-bearing paths and choose enforcement placements accordingly. Practical placements include egress gateways, retrieval gateways, service-mesh sidecars, key-release gates, update-channel gates, message brokers, and recipient-side verifiers.

Before using ORPRG for production decisions, operators need an implementation-specific threat model, key and issuer governance, monitoring, failure-mode analysis, and operational recovery procedures.

18. Implementation Status

This section records implementation status for IETF discussion and is expected to be updated or removed before RFC publication in accordance with the guidance in [RFC7942].

A synthetic reference evaluation artifact exists for the ORPRG verifier concept. It exercises deterministic JSON canonicalization, action digests, synthetic signatures, strict schemas, status and checkpoint checks, Merkle-style inclusion and non-inclusion proofs, capability-token validation, replay caches, retrieval and egress gateway demonstrations, an ext_authz-shaped adapter, a KMS/HSM-shaped key-release gate, HTTP-envelope canonicalization fuzzing, conformance vectors, and baseline matrices. The artifact is synthetic and is not production code.

The artifact is intentionally separated from this draft so that the community can discuss the protocol, conformance expectations, and appropriate public artifact release posture independently of any particular research package.

19. Acknowledgments

The author thanks the IETF security, identity, attestation, transparency, and authorization communities for the mechanisms that make interoperable effect-boundary authorization possible.

20. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", 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", RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

21. Informative References

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

[RFC7942] Sheffer, D. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC5280] Cooper, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6960] Santesson, S., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[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>.

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

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9449] Fett, D., "OAuth 2.0 Demonstrating Proof of Possession", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

[RFC8693] Jones, M., Nadalin, A., and B. Campbell, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/info/rfc8693>.

Author’s Address

Yong Bok Lee
Meridian Verity Group
Sheridan, WY
United States of America
Email: scott@meridianverity.com