GRIP: The Governed Runtime Intent Protocol

An open protocol for authorised, bounded and attributable agent execution

— Request for Comments, draft.2

Madhav D. Parashar (Chief Author) · linkedin.com/in/madhavdutta

Arnav Parashar (Co-Author) · linkedin.com/in/arnavparashar49

Foxora Labs Ltd · foxora.ai

Draft v2.1 — Request for Comments. August 2026. Protocol version GRIP/1.0-draft.2. This document is issued for public comment and adversarial review; it is not a frozen specification. The specification artifacts — JSON Schemas, registries, verifier, and conformance suite — are released with this paper under an open license.

Abstract

Agentic AI systems act — they call tools, spend money, message people, and change production state — yet the protocols connecting them standardize only *capability*: what an agent can reach (MCP), and how agents find and message one another (A2A). Neither expresses the governance of an execution — who authorized it, under what limited and expiring authority, within which budget, subject to whose approval, and provable to which auditor. (The Agent Payments Protocol expresses signed authorization and settlement, scoped to commerce.) This gap is independently documented in the literature and is being closed under regulatory pressure: a cluster of 2026 proposals now reaches the governance floor, and §2 scores them against us candidly — one of them matches GRIP on every column of our rubric. We propose **GRIP** (the Governed Runtime Intent Protocol), a small, open protocol that makes agent work **authorised before, bounded at an enforcing gate, and cryptographically attributable and integrity-verifiable afterwards** — regardless of model, framework, or vendor. GRIP is six signed JSON objects on one hash chain — **Mandate, Grant, Receipt, Stop, Settlement, and the Context Envelope** — with a five-family act registry (`tool.*`, `model.*`, `agent.*`, `skill.*`, `stop.*`) and a typed denial-reason registry. Four conformance levels let an implementer start by emitting verifiable receipts and grow into full evidence-gated settlement. Authority only narrows as it delegates; every admission *denial* is a typed, prose-carrying refusal; human approval is an attributable, act-bound decision the runtime cannot fabricate under the signature approval mode; and budget charges land on the same chain as the acts they meter. GRIP composes with rather than competes against the existing stack: it scopes MCP tools, meters model calls, delegates to A2A agents, and admits open skill packages — one gate, one receipt shape. Because its schemas compile to decoding grammars, its denials are typed, and its settlements are outcome labels, GRIP is also designed to be *trained against*, defining a native interface for contract-native models (Agentic Language Models). We give the normative protocol, a threat model with security arguments and named residual risks, a scored coverage analysis against a published governance-gap taxonomy, and a reference implementation released with this paper.

---

1. Introduction

The agentic era inverted a safety assumption. When language models only wrote text, the human between the model and the world was the governance layer. Agentic systems removed that human from the inner loop: models

now invoke tools, purchase inference, spawn sub-agents, and commit side effects at machine frequency. The connective protocols that enabled this — and they are genuinely good protocols — answer *capability* questions. The Model Context Protocol [1] standardizes how an agent reaches tools and data, including an authorization framework for the client–server connection. Agent2Agent [2] standardizes how agents discover and message one another, with an extension mechanism for what cards and tasks may carry. Neither, by design, answers the questions an operator, a security team, or a regulator asks:

- **Who authorized this action, and does the authorization chain reach a human?**
- **What was this agent permitted to do, for how long, and how was the permission bounded below its issuer's?**
- **What did the work cost, against what agreed budget?**
- **Which actions required a human decision, and can the system prove the human — not the system — made it, about this act and not another?**
- **When the work is "done," done by whose criteria, checked by whom, and provable to a verifier outside the operator's own trust domain?**

The absence is documented, not conjectural. A 2026 evaluation of five interoperability protocols against a six-dimension governance taxonomy finds no protocol carrying the governance dimensions, and concludes that community governance in particular is "a missing architectural layer above current interoperability standards" [6]. An audit framework applied to six popular open-source agent stacks reported 617 findings and argued that no agent system can be accountable without auditability [7]. The AIP proposal [8] — a delegation protocol retrofitted across MCP and A2A — was motivated by a survey of roughly two thousand deployed MCP servers that found them uniformly unauthenticated; the survey predates MCP's 2026-07-28 authorization hardening and measures deployment practice, not specification capability, but the practice is the point. Meanwhile the reliability literature shows why enforcement cannot be delegated to model quality: on τ-bench's retail domain, GPT-4o passed 61% of tasks once but all eight of eight attempts only about a quarter of the time [15], and long-horizon agents derail in ways uncorrelated with context exhaustion [17]. Regulation is arriving on a fixed clock: the EU AI Act's Article 12 record-keeping obligations for high-risk systems apply from 2 December 2027 under Annex III and 2 August 2028 under Annex I, as deferred by the Digital Omnibus [26].

The research community's response to date has been what we call **exoskeletal**: capability firewalls [10], privilege-control DSLs [11], proof-carrying action checkpoints [12], receipt notaries [13], and deontic policy engines [14] — each a sound mechanism, each wrapped around an unmodified agent. The defect is not the wrapping — GRIP's own lowest conformance levels wrap unmodified agents too. The defect is fragmentation: no two systems share contract types, so no receipt verifies across systems, no authority vocabulary transfers, and no training signal accumulates. Protocols fix exactly this class of problem, and the bar for a successful one is known: small enough to hold in one head, adoptable incrementally, built from boring cryptography, and silent about implementation. HTTP did not specify the web server; OAuth did not specify the database. GRIP does not specify the agent runtime.

Contributions.

1. **The protocol** (§4): six signed, content-addressed JSON objects on one hash chain — Mandate, Grant, Receipt, Stop, Settlement, Context Envelope — with normative state machines, a defined attenuation (dominance) order, an eight-rule verification algorithm, an act registry spanning tools, models, agents, skills, and protocol control, and a typed denial-reason registry.

2. **An adoption ladder** (§4.7): four conformance levels, from GRIP-0 (emit verifiable receipts) to GRIP-3 (evidence-gated settlement with independent validation).
3. **Composition** (§5): precise bindings to MCP (both directions), A2A, open skill packages, external model calls, telemetry, and payment protocols — one gate and one receipt shape across all of them.
4. **A threat model and security argument** (§6): authority provenance and designated issuance, attenuation-only delegation, act-bound single-use approvals, TTL containment, chain anchoring, single-door injection containment — with residual risks named, including the ones that survive.
5. **The trainable-protocol property** (§7): a governance protocol designed so that models can be made *native* to it — schemas that compile to decoding grammars, typed denials as labelled negatives, settlements as outcome labels — defining the interface for Agentic Language Models (ALMs). Every individual mechanism in GRIP has prior art (§2); the contribution is their composition into one model-facing object model.
6. **A reference implementation and artifacts released with this paper** (§8): an engine targeting the GRIP-3 semantics, SDK middleware that upgrades existing agents to GRIP-0 without rewrite, a standalone verifier, and the conformance suite.

2. Related work

Interoperability protocols. MCP [1] (spec revision 2026-07-28) provides stateless capability exchange between clients and tool/resource servers, with connection-level authorization (RFC 9207 issuer validation, client-identity metadata) and an elicitation/`input_required` flow — a protocol-level pause-for-human hook, though without approval semantics: no authority binding, no proof of who approved, no typed rejection, no consequence classing. GRIP's Stops supply exactly those semantics (§4.3.4). A2A [2] (v1.0.1, Linux Foundation) provides agent discovery via Agent Cards and a task lifecycle between opaque agents; its v1.0.1 extension mechanism is the natural carrier for the conformance-level advertisement GRIP proposes (§5.2). AGNTCY/ACP [3] added identity, discovery and invocation schemas; the ACP specification was archived read-only on 11 April 2026 at v0.2.3, and AGNTCY now carries A2A over its own SLIM transport while emphasising Directory, OASF, Identity and Observability; GRIP's trust model (§4.1) consumes precisely that layer rather than reinventing it. IETF Web Bot Auth [4] puts per-agent signatures on HTTP requests — *which bot*, not *what it may spend*. The Agent Payments Protocol [5] is the closest philosophical relative — signed, verifiable mandates and settlement for agent purchases, scoped to commerce; §5.5 treats it as an extension surface, not an alternative.

Capability tokens and attenuated delegation. GRIP's Grants stand on a well-developed line: Macaroons [31] established attenuation-only caveated bearer tokens with TTLs and no revocation; Biscuit [32] added offline attenuation with datalog policies; UCAN and ZCAP-LD carry capability chains in decentralized identity systems; OAuth's Rich Authorization Requests (RFC 9396 [33]) and token exchange (RFC 8693) bring fine-grained, delegable authorization to the OAuth family. GRIP adopts this line's core discipline — attenuation-only, expiring, offline-verifiable authority — and adds what no token format carries: the surrounding contract (Mandates with acceptance criteria and budgets), the evidence layer (receipt chains, settlements), typed denials, human approval objects, and the model-facing context envelope. A Grant is a capability token that knows what it is *for* and must answer for what it *did*.

Governance gap analyses. Kang & Diponegoro [6] evaluate MCP, A2A, ACP, ANP and ERC-8004 against a six-dimension *community* governance taxonomy — membership, deliberation, voting, dissent preservation, human escalation and audit/replay — and find the governance layer absent across all five. Their instrument asks how a

collective of agents decides; ours asks how a single execution is authorised and proved. Table 1 is therefore a distinct, author-defined **execution-governance rubric** derived from the five operator questions of §1 — not their taxonomy, which we score separately and partially in §9. Their per-dimension findings — and AIP's [8] survey motivation and Auditable Agents' [7] 617 findings across six stacks — ground the gap claim of §1. Auditable Agents also demonstrates affordability: median 8.3 ms mediation overhead for tamper-evident pre-execution records. AIP itself [8] delivers attenuated, verifiable delegation across MCP and A2A via invocation-bound capability tokens at 0.22–2.35 ms overhead — genuine functional overlap with GRIP's Grants. GRIP differs in scope: AIP is a delegation layer; GRIP binds delegation to mandates, budgets, stops, receipts, and settlement on one verifiable chain, and is model-facing (§7).

Runtime governance mechanisms. CaMeL [10] separates privileged from quarantined models with capability-tracked data flow; Progent [11] enforces least-privilege tool policies with SMT-checked narrowing; proof-carrying agent actions [12] mediate five checkpoints around any model; Sello [13] has counterparties sign receipts of observed agent calls — an anchoring idea GRIP adopts for settle-class acts (§4.6); deontic runtime policies [14] formalize obligation and permission at run time. GRIP's relation to this line is that of a wire standard to mechanisms: each could emit and consume GRIP objects, gaining interoperable receipts and a shared authority vocabulary without abandoning its enforcement approach.

Governance protocols for agent execution — the closest prior art. Every mechanism in GRIP has prior art, and we claim none of them individually. **OpenPort** [37] specifies governance-first tool access with a draft-first write lifecycle, *preflight impact binding* that hashes the tool, payload and computed impact before approval, state-witness preconditions revalidated at execution, least-privilege scopes, and a stable `agent.*` reason-code registry with an executable conformance toolchain — the closest existing work to our act-bound Stop and typed-denial registry. **Intent-Governed Tool Authorization** [38] formalises attenuation as *monotonic narrowing*, proving a session policy is never more permissive than the static policy it refines, and maintains a tamper-evident local audit sequence; its certificates are server-issued rather than human-signed, and carry effect bounds rather than spend budgets. **SessionBound** [39] issues signed, human-approved, budget-carrying session tokens with TTLs, nonces and revocation, and enforces query and disclosure budgets at the database layer. **Foundation Protocol** [40] is the closest of all: an open coordination layer whose contract snapshots form an explicit hash chain, whose sessions carry spend limits and token ceilings, whose checkpoints defer high-value decisions to a human owner, and whose arbiter escrows funds until delivery acceptance and then issues a signed receipt — policy, provenance, metering, receipts, settlement and oversight as protocol concerns, with a released runtime.

Against that record, the honest statement of contribution is narrow. GRIP does not introduce signed intent, attenuated capabilities, spend budgets, human approval gates, hash-chained receipts, typed refusals or evidence-gated settlement. What we have not found elsewhere is the particular composition: one interoperable object model in which a human-signed mandate, attenuating grants, act-bound approval, a receipt chain and evidence-gated closure are joined to a **model-facing context envelope**, with conformance levels that let a system adopt the weakest useful part first — and with the whole set designed to be a training interface as well as an audit surface (§7). Readers should weigh that composition claim against [37]–[40] directly.

Delegation and identity. Authenticated delegation from humans to agents via OAuth/OIDC extensions is developed in [9]; workload identity (SPIFFE [30]) supplies attested per-workload keys. Any identity system that can publish a verification key can sign GRIP objects.

Reliability and the case for external bounds. τ-bench and τ^2-bench [15, 16] established pass^k collapse in policy-constrained agents; Vending-Bench [17] documented long-horizon derailment uncorrelated with context

size. GRIP's bounds hold at the gate regardless of the policy's quality.

Small and contract-trained models. The position that small models suffice for most agentic invocations is argued in [18]. Trained orchestrators demonstrate small models directing tools and stronger models: ParaManager [20] trains under a protocol-compliance reward and Orchestrator-8B [19] under an outcome/cost/preference reward, while HALO [21] trains on verifier-certified trajectories for planning validity rather than tool routing. When2Call [22] and BAGEN [23] train fragments of governance behavior (abstention, budget feasibility). To our knowledge, none of these systems trains against a governance contract — no published contract has existed to train against. §7 develops this connection; a companion paper develops the model class.

Table 1 — What the protocols express (per-cell grading against the published specifications; ● expressed, ◐ partial, ○ absent):

	Authority chains to a human	Attenuated delegation	Budgets metered	Human approval semantics	Settlement w/ evidence	Third-party verifiable record
MCP [1]	○	○	○	◐ (elicitation hook)	○	○
A2A [2]	○	○	○	○	○	○
ACP/AGNTCY [3]	◐ (identity)	○	○	○	○	○
Web Bot Auth [4]	◐ (signed requests)	○	○	○	○	◐
AP2 [5]	● (commerce)	○	◐ (payment caps)	◐ (mandate consent)	● (commerce)	● (commerce)
AIP [8]	●	●	○	○	○	◐
OpenPort [37]	○	●	◐ (quotas)	● (preflight-bound)	○	◐ (events, no chain)
Intent-Gov. Tool Auth. [38]	○	● (monotonic)	○ (effect bounds)	●	○	◐ (local chain)
SessionBound [39]	●	◐	●	●	○	◐ (receipts)
Foundation Protocol [40]	●	●	●	● (owner checkpoints)	● (escrow)	● (hash chain)
GRIP/1.0-draft.2	●	●	●	●	●	● (§6 limits named)

Read this table against us, not for us. The Foundation Protocol [40] scores as we do on every column, and three of the other four match or beat GRIP on individual mechanisms — OpenPort's preflight impact binding is a stronger act-to-approval binding than our own first draft carried. That is the correct result: the rubric measures the *governance floor*, and others have reached it. What it does not measure, and what the rest of this paper argues is the difference, is the model-facing Context Envelope as a first-class protocol object (§4.3.6), the conformance ladder that lets a system adopt the weakest useful part first (§4.7), and the design of the whole object model as a training interface rather than only an audit surface (§7). A reader who concludes from this table that the governance floor is now a crowded field has understood it correctly.

3. Design principles

P1 — Adoptable in an afternoon at the lowest level. A protocol that demands a runtime is a product, not a protocol. GRIP-0 is a few lines of middleware with the SDK; "an afternoon" is the design target the released examples are measured against (§9).

P2 — Boring cryptography, borrowed identity. JSON canonicalized with JCS (RFC 8785 [27]), signed as detached JWS with Ed25519 and an unencoded payload (RFC 7515/8037/7797 [28, 34]), keys discovered via JWKS URLs or DID documents. GRIP introduces no novel cryptography and no identity system.

P3 — Messages and rules, never runtimes. The specification defines object shapes, state machines, and verification rules. It says nothing about queues, schedulers, memory, storage, model routing, or context *selection*. It does impose two rendering constraints on trust-classed content (§4.3.6) — the minimum without which the Context Envelope's security meaning would be void — and marks them as conformance-clause obligations rather than chain-verifiable rules.

P4 — Content addressing everywhere. Every object's `id` is the hash of its canonical form. References are verifiable, replay of an object into a foreign chain is detectable, and identical requests deduplicate.

P5 — Authority chains to a designated human root, and only narrows. Every authority-bearing object traces, by hash and by *designated issuance* (§4.3.1), to a human-signed Mandate. Grants attenuate under a defined dominance order; nothing in the protocol can widen authority — including, deliberately, skills and delegated agents. (Named exception: GRIP-0 witness chains may be mandate-less; §4.7.)

P6 — Time bounds instead of revocation infrastructure. All authority expires; nothing is long-lived enough to need a revocation list in v1. A revocation extension is reserved and `crit`-marked (§4.8, §10).

P7 — Fail closed at gates, must-ignore at envelopes. An enforcement point that cannot evaluate a request denies it, with a typed reason. A parser that meets an unknown field ignores it unless the field is listed in the object's `crit` array. Budget axes are the stated exception: a runtime that cannot meter an axis present in a governing budget must refuse acts under it (§4.5).

P8 — Every denial is typed. Refusals carry machine-readable reason codes from a public registry; the implementation-defined escape code requires a namespaced subcode, and conformance reporting surfaces its share of denials so registry gaps become visible pressure (§4.4).

4. The GRIP protocol

The key words MUST, MUST NOT, REQUIRED, SHALL, SHOULD, SHOULD NOT, MAY and OPTIONAL in this section are to be interpreted as described in BCP 14 (RFC 2119, RFC 8174) when, and only when, they appear in all capitals.

4.1 Actors and trust model

- **Principal** — a human or organization on whose behalf work happens; the root of all authority.
- **Agent** — any acting software: a model-driven runtime, a framework loop, a script. Identified by a published key.

- **Runtime** — the enforcement point that evaluates acts against Grants and emits Receipts. Agent and runtime may be one process (self-reporting; honest at GRIP-0/1) or separated (enforcing, from GRIP-2).
- **Approver** — a human identity a Stop designates; may be the Principal.
- **Verifier** — any party, inside or outside the operator's trust domain, that checks objects and chains after the fact.

Trust derives from signatures. Each deployment publishes a **trust bundle** at `https://{origin}/.well-known/grip`: one signed JSON document enumerating its issuer identities, their JWKS or DID references, their roles, and validity windows, signed by the deployment's bundle key (rooted in the origin's TLS identity, with external attestation as a reserved extension). The trust bundle is the only thing a third-party verifier needs to fetch.

4.2 The envelope

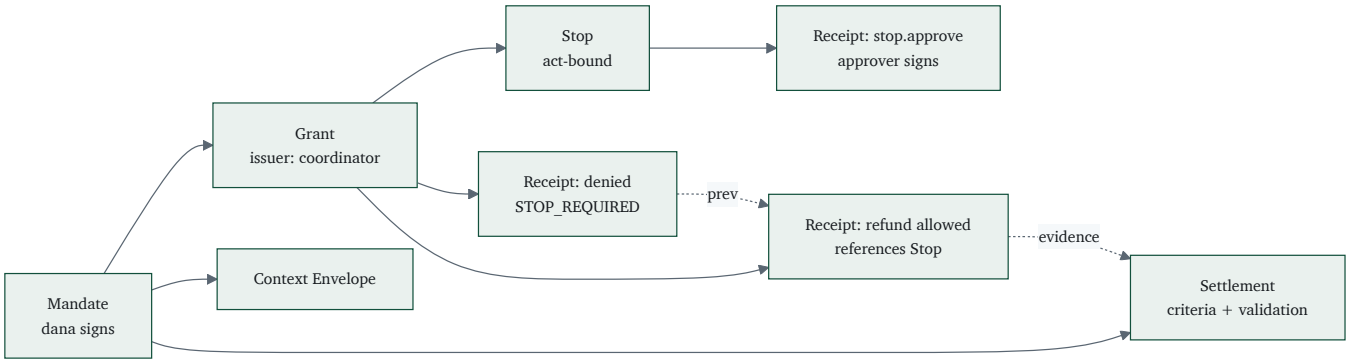
Every GRIP object shares one envelope:

Field	Meaning
`grip`	protocol version, e.g. `"1.0-draft.2"`; a released version drops the suffix
`type`	`mandate` | `grant` | `receipt` | `stop` | `settlement` | `context`
`id`	`sha256:` + hash of the JCS-canonical object with `id` and `sig` absent
`parents`	hash references to the objects this one derives from; absent on a Mandate (the root) and on GRIP-0 witness Receipts
`prev`	**Receipts only**: hash of the previous Receipt in the same chain; absent on the first Receipt
`iss`	issuer identifier (HTTPS origin key reference or DID)
`iat`, `exp`	issued-at and expiry, integer seconds. `exp` is **required** on the authority-bearing types — Mandate, Grant, Stop — and optional elsewhere
`crit`	optional array of extension field paths (slash-delimited for nesting, e.g. `"budget/co2_mg"`) a consumer **must** understand to process this object
`sig`	detached JWS, EdDSA, protected header exactly `{"alg":"EdDSA","kid":<key reference matching` iss `>,"b64":false,"crit":["b64"]}` — RFC 7797 requires `b64` to be integrity-protected and listed in `crit` — unencoded detached payload = the JCS-canonical object with `id` and `sig` absent (the same preimage as `id`), compact serialization

Timestamps are issuer-asserted; §4.6 rule 8 and the anchoring mechanism bound what dishonest clocks can achieve. Canonical representation is JSON/JCS; a CBOR/COSE profile is reserved.

4.3 The six objects

Figure 1 — the object graph (one Mandate's chain, the worked example of Appendix C):



4.3.1 Mandate — the agreement

A Mandate is the Principal's signed statement of what is wanted, what it may cost, and **who may issue authority against it**.

```
{ "grip": "1.0-draft.2", "type": "mandate",
  "id": "sha256:3de8…",
  "iss": "https://id.acme.com/people/dana#key1",
  "goal": "Resolve open billing tickets, refunds under $50 each",
  "constraints": ["never contact the customer's bank",
                  "refunds only on tickets tagged verified"],
  "scope": [
    { "verb": "tool.call",  "resource": "mcp://crm/*" },
    { "verb": "model.call", "resource": "class:reasoning" } ],
  "issuers": ["https://runtime.acme.com/issuers/coordinator#key3"],
  "acceptance": [
    { "code": "tickets-closed", "check": "all selected tickets in state closed" },
    { "code": "refund-ledger",  "check": "every refund posted with ticket reference",
      "independent": true } ],
  "budget": { "usd_micro": 200000000, "tokens_in": 5000000,
              "tokens_out": 1000000, "ms": 86400000 },
  "iat": 1787000000, "exp": 1787086400, "sig": "…" }
```

Normative rules. `goal`, at least one `acceptance` entry, `budget`, and `issuers` are required. `issuers` names the key identities permitted to sign root Grants against this Mandate; a root Grant from any other issuer is a verification failure (§4.6 rule 3); membership in the trust bundle alone never suffices to issue against a Mandate. `scope` is an optional but strongly recommended machine-checkable authority ceiling in Grant-`allow` shape; when present, root Grants are checked against it mechanically. `goal` and `constraints` are human context and settlement-time evidence — they bind the *issuer's accountability*, not the verifier's algorithm; verifiers report each chain's root hop as **machine-checked** (scope present) or **issuer-attested** (scope absent), the same way they report hard-budgeted versus unmetered. A budget axis that is absent is unmetered; a Mandate is **hard-budgeted** only when all four axes are present. The Mandate's state machine is `open → settled | cancelled`, with state recorded by Settlement objects (§4.3.5), never by mutation — GRIP objects are immutable once signed. A

cancelling Settlement may be signed by the Principal or by an issuer the Mandate lists; after a Mandate's terminal Settlement, its Grants authorize nothing (§4.6 rule 6).

4.3.2 Grant — the permission

A Grant conveys attenuated authority from a Mandate (or, when delegable, from another Grant) to an actor.

```
{ "grip": "1.0-draft.2", "type": "grant",
  "id": "sha256:75e3…", "parents": ["sha256:3de8…"],
  "iss": "https://runtime.acme.com/issuers/coordinator#key3",
  "actor": "https://agents.acme.com/support-bot#key2",
  "allow": [
    { "verb": "tool.call",  "resource": "mcp://crm/*",      "class": "observe" },
    { "verb": "tool.call",  "resource": "mcp://crm/refund", "class": "settle"  },
    { "verb": "model.call", "resource": "class:reasoning",  "class": "egress",
      "boundary": ["no-public-cloud"] } ],
  "budget": { "usd_micro": 60000000, "tokens_in": 500000,
              "tokens_out": 100000, "ms": 3600000 },
  "iat": 1787000100, "exp": 1787003700, "sig": "…" }
```

Every `allow` entry **must** carry a `class` (§4.4); there is no default. Where entries overlap — the observe-class wildcard above also matches `mcp://crm/refund` — **the most consequence-severe matching class governs** (settle > egress > observe), so a settle listing can never be bypassed through a broader observe pattern.

The attenuation laws. Between a child Grant and its parent (Mandate `scope` at the root hop when present; parent Grant `allow` otherwise), the child must be **dominated**:

1. *Entry dominance.* Every child `allow` entry must be dominated by some parent entry: `verb` equal; child `resource` contained in the parent's resource pattern (a trailing `*` matches any completion of the prefix; exact strings match only themselves; no other wildcard forms in v1); child `class` equal to or more restrictive than the parent's for that resource (observe may narrow to settle — adding a Stop requirement is a restriction — never the reverse; egress may narrow to settle); child `boundary` a superset of the parent's (constraints are added, never dropped).
2. *Budget dominance.* The child must carry **every axis present in its parent**, each ≤ the parent's remaining value; an axis absent from both is unmetered; a child may add axes. For comparison, absent-in-parent means unbounded. Issuing a Grant **reserves** its budget against the parent's remaining axes; sibling Grants compete for the reservation (§4.5).
3. *Expiry dominance.* `child.exp ≤ parent.exp`.
4. *Delegation off by default, and issuance is authorised.* A Grant may parent another Grant only if it carries `"delegable": true`; the child obeys laws 1–3 against it. Attenuation alone is not sufficient: the child's `iss` MUST be the parent Grant's `actor`, or a key the parent lists in `delegation_issuers`. A runtime may issue on an actor's behalf only by including that actor's detached signature over the child Grant's canonical form. Verifiers reject a delegated Grant whose issuer holds no such authority — dominance without issuance authority is not delegation, it is minting.
5. *Wildcard confinement.* `*` patterns may appear only in root Grants signed by a Mandate-listed issuer — never in delegated Grants.

At GRIP-2 and above, an act request must be signed by the Grant's `actor` key, and the Receipt carries that signature (`act.actor_sig`, a detached JWS by the actor key over the grant id and the canonical act) — making actorship third-party-verifiable and a leaked bare Grant unusable without the actor's key.

4.3.3 Receipt — the record

One act, one decision, one Receipt, hash-linked to the previous Receipt in the chain.

```
{ "grip": "1.0-draft.2", "type": "receipt",
  "id": "sha256:f64b…",
  "parents": ["sha256:3de8…", "sha256:75e3…"],
  "prev": "sha256:14a4…",
  "iss": "https://runtime.acme.com/issuers/gate#key1",
  "actor": "https://agents.acme.com/support-bot#key2",
  "act": { "verb": "tool.call", "resource": "mcp://crm/refund",
           "input_digest": "sha256:f97b…", "actor_sig": "…" },
  "context": "sha256:1bbd…",
  "decision": { "result": "denied", "reason": "GRIP_DENY_STOP_REQUIRED" },
  "cost": { "usd_micro": 0, "tokens_in": 0, "tokens_out": 0, "ms": 4 },
  "iat": 1787000230, "sig": "…" }
```

Normative rules. `decision` is `{result: "allowed"|"denied", reason?, stop?, crit_object?}`: a denial **must** carry a registry `reason`; `stop` references the governing Stop where one applies; `crit_object` names the object and extension path when the reason is `GRIP_DENY_UNKNOWN_CRIT`. `context` references the Context Envelope a model-driven act was decided from (required at GRIP-3; §4.6 rule 7). At GRIP-0/1 a single Receipt self-reports decision and outcome together. At GRIP-2/3, where the gate decision and the world effect are separated in time, the allowed act produces a **paired outcome Receipt**: it carries `gate: <gate-receipt id>`, occupies its own `prev` slot, and its `outcome` object records `{status: "ok"|"failed"|"interrupted", effects: [...], output_digest, resolved?: {...}}`. Accounting: the gate Receipt's `cost` is the reserved estimate and binds only for pre-act denial; the outcome Receipt's `cost` is actual and **supersedes** its paired estimate in all recomputation (§4.6 rule 4). Denied Receipts charge their recorded (gate) cost. Inputs and outputs appear as digests, never bodies. Bare digests are integrity bindings, **not** confidentiality: low-entropy inputs are dictionary-recoverable from their hashes, so implementations handling sensitive arguments SHOULD use salted commitments (`sha256(salt ‖ input)` with the salt escrowed under the deployment's disclosure policy), and the specification says so rather than implying secrecy digests do not provide.

Receipts within one Mandate's chain are serialized: GRIP/1.0 requires a single chain-head authority per Mandate (the gate named by the Mandate's issuer set), and concurrent acts serialize through it. A delegated segment (§5.2) forms a sub-chain rooted at the delegating Grant, reported back by settlement-style receipt; DAG chains are reserved for GRIP/2.0.

4.3.4 Stop — the human gate

A Stop suspends a specific act until a designated human decides. It is **act-bound and single-use**: the Stop carries the digest of the act it gates, and at most one allowed settle-class act may reference an approved Stop. (A Stop may instead declare an explicit multi-use envelope — `{count, budget}` — which the approval surface must display; absent that declaration, one approval is one act.)

```
{ "grip": "1.0-draft.2", "type": "stop",
  "id": "sha256:2f98…", "parents": ["sha256:3de8…", "sha256:75e3…"],
  "iss": "https://runtime.acme.com/issuers/gate#key1",
  "act_digest": "sha256:f97b…",
  "reason": "settle-class act: refund $42 on ticket 8812",
  "approver": "https://id.acme.com/people/dana#key1",
  "mode": "signature",
  "iat": 1787000230, "exp": 1787004000, "sig": "…" }
```

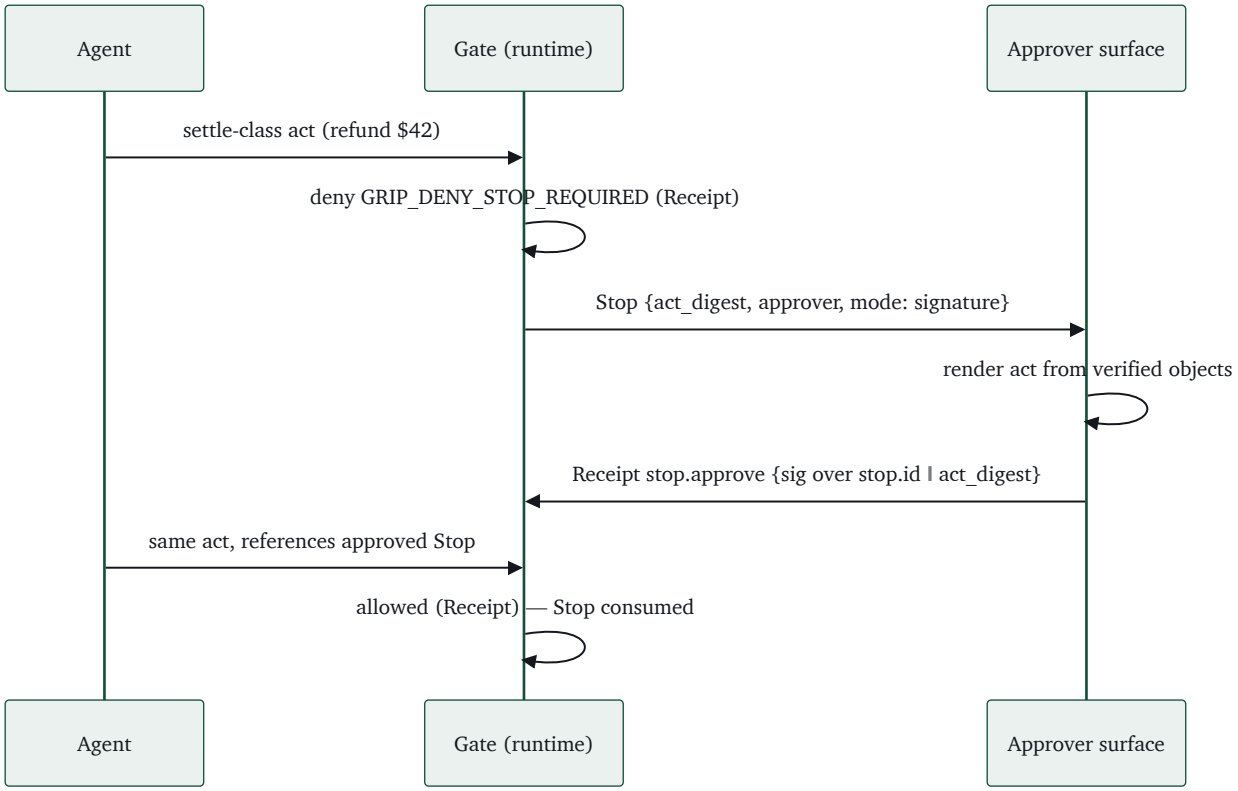
Two approval modes, declared by the Stop:

- **`signature` (preferred).** The approval is a Receipt with verb `stop.approve` whose `approval` field is the approver's detached signature over `(stop.id ‖ act_digest)`, verified against the approver's published key. No secret ever exists for the runtime or any delivery path to leak: **the runtime structurally cannot fabricate this approval**, and the property holds against the runtime itself, not merely the acting agent.
- **`token` (fallback, for approvers without keys).** The runtime mints a random 32-byte completion token, stores only its SHA-256 digest in the Stop, and delivers the token out-of-band; approval discloses the token, consumed single-use. This mode's trust is stated honestly: the runtime generated the token and is *trusted to discard it*, and every hop of the delivery path holds a bearer credential — fabrication resistance is against a runtime that kept its promise, which is policy, not structure. Deployments that can give approvers keys should.

The Stop is the approver's capability. A `stop.approve` or `stop.reject` Receipt is issued by the named `approver`, who ordinarily holds no Grant. Rather than exempt control acts from §4.6 rule 3, the Stop itself is the one-shot authority for exactly two acts by exactly that approver, and rule 3 is satisfied by the Stop in place of a Grant chain. No other act may be authorised this way.

Normative rules for both modes: verification order is expiry first, then approval check (constant-time for token digests), so a lapsed Stop is not an oracle; the approval must be attributable to the Stop's named `approver`; a rejection (`stop.reject`) **must** carry a reason code; an expired Stop resolves to `expired`, never approval. **Approval-surface conformance clause:** a conformant approval surface renders the Stop, its bound act digest, and the act's human-legible description *from verified chain objects*, never from runtime-supplied prose alone — otherwise the human approves a caption, not an act. Raising a Stop is itself a governed act (`stop.request`, §4.4): grantable, receipted, and rate-disciplined, so approval-fatigue flooding is a visible, deniable behavior rather than a free one.

Figure 2 — the Stop flow:



4.3.5 Settlement — the closing

A Settlement closes a Mandate against its own acceptance criteria.

```
{ "grip": "1.0-draft.2", "type": "settlement",
  "id": "sha256:1a5e…", "parents": ["sha256:3de8…"],
  "iss": "https://runtime.acme.com/issuers/settle#key4",
  "state": "settled",
  "criteria": [
    { "code": "tickets-closed", "verdict": "accepted",
      "evidence": ["sha256:f64b…", "sha256:04a7…"],
      "validator": "https://runtime.acme.com/issuers/settle#key4" },
    { "code": "refund-ledger", "verdict": "accepted",
      "evidence": ["sha256:1f3a…"],
      "validator": "https://id.acme.com/people/lee#key1",
      "validator_sig": "…" } ],
  "totals": { "usd_micro": 42480000, "tokens_in": 413000,
              "tokens_out": 21100, "ms": 2399990 },
  "iat": 1787006000, "sig": "…" }
```

Normative rules. Every acceptance criterion of the Mandate must be addressed with a verdict and evidence references (digests into the Receipt chain and to external artifacts). A criterion the Mandate marks `"independent": true` must be validated by an identity distinct from the acting agent, and that validator **signs its verdict** (`validator_sig`, detached JWS over the criterion entry) — without a validator signature,

independence would be the settlement issuer's unverifiable assertion. v1's independence predicate is identity inequality; distinct-principal and distinct-organization predicates are profile extensions, and verifiers report which predicate each settlement satisfies. An objection — a signed rejection verdict by any Mandate-listed issuer or the Principal, recorded as a `rejected` criterion entry — blocks a `settled` state until superseded; nothing is averaged away. `state` is `settled` or `cancelled`; who may cancel is defined in §4.3.1. `totals` restate the chain's recomputed spend for auditor convenience and must match §4.6 rule 4's recomputation. In the published vector that total is `42480000` — a $42.00 refund plus $0.48 of inference fees — which is §4.5's one-axis question in concrete form: the fee paid to do the work and the value moved by the work land on the same axis, and an auditor reading `usd_micro` alone cannot separate them.

4.3.6 Context Envelope — what the model saw

The Context Envelope is GRIP's model-facing object: a signed record of the exact input a model-driven agent acted on.

```
{ "grip": "1.0-draft.2", "type": "context",
  "id": "sha256:1bbd…", "parents": ["sha256:3de8…"],
  "iss": "https://runtime.acme.com/issuers/compiler#key2",
  "purpose": "select next act for ticket batch 7",
  "budget_tokens": 4000, "used_tokens": 3112,
  "items": [
    { "ref": "mandate", "digest": "sha256:3de8…", "trust": "principal" },
    { "ref": "crm-history-8812", "digest": "sha256:a549…", "trust": "external" },
    { "ref": "skill:billing-runbook@sha256:cb51…", "digest": "sha256:cb51…",
      "trust": "governed" } ],
  "iat": 1787000200, "sig": "…" }
```

Trust classes are normative: **`principal`** — content authored or signed by the Principal (the Mandate itself, their direct instructions); **`governed`** — content an operator has reviewed and pinned by digest, with receipted provenance; **`external`** — everything else, including tool results, model replies, retrieved documents, and accreted stores of past external input (a CRM history is `external` no matter whose database it sits in). **Taint propagation:** a derived or stored item inherits the *lowest* trust class among its sources, permanently, unless a receipted declassification act with a named authority upgrades it — storage laundering is thereby a chain-visible event, not a default. Rendering constraints (conformance-clause obligations; not chain-verifiable, and the spec says so): instruction-privileged channels admit only `principal`-classed content; `governed` content is framed and attributed but carries no instruction privilege; `external` content is delimited by an explicit untrusted-content frame. Skill scripts execute under their package's trust class and their emissions are derived content under the taint rule. The envelope is token-budgeted (`used_tokens ≤ budget_tokens`), and a model-driven act's Receipt references the envelope it was decided from (`context` field) — which turns prompt-injection response from forensics into lookup. How an implementation *selects* context is unspecified and competitive.

4.4 The act registry

Acts are verbs applied to resources, in five families — one gate, one Receipt shape:

Family	Meaning	Resource examples
`tool.*`	effects on and observations of the world	`mcp://{server}/{tool}`
`model.*`	purchases of cognition	`class:reasoning`, `class:embedding`
`agent.*`	delegation to autonomous agents	A2A card URL, GRIP runtime URL
`skill.*`	loading packaged procedures	skill package digest
`stop.*`	protocol control: `stop.request`, `stop.approve`, `stop.reject`	a Stop id

Consequence classes partition acts three ways: **`observe`** (read-like, no world mutation), **`egress`** (no world mutation, but externally visible — data can leave through it: outbound messages, external searches, model calls to outside providers), **`settle`** (world-changing; requires an approved Stop). Classing is assigned by the Grant issuer; because misclassification is not chain-detectable, verifiers report every resource that executed settle-free, and the residual risk is named in §6.

`model.call` — models as governed workers. The Grant names a model *class* and a data `boundary` set from the released registries, never a vendor endpoint; the runtime resolves the concrete model and redeems credentials at its own trusted edge — the acting agent never holds provider credentials. The **outcome Receipt must record the resolved provider and model identifier and the boundary it was matched under** (`outcome.resolved`), so boundary compliance is chain-checkable rather than a configuration claim. Cost lands on the Receipt; the reply re-enters as `external` context. One honest limit, stated in §6: a model wrapped as an MCP tool evades `model.*` metering — Grants can attach boundary labels to tool resources for exactly this case, but the partition is ultimately as good as the Grant author's inventory.

`skill.load` — skills as content, never authority. A skill package is identified by digest and version; loading is a receipted act; its text enters the Context Envelope under its provenance class and the taint rule; and **nothing a skill contains can widen a Grant** — a skill may use granted authority, never mint or escalate it. Pinning is provenance, not review: a pinned package's contents are `governed` only if the operator's review process says so, and its scripts' runtime emissions are derived content regardless.

The denial-reason registry (Appendix A) ships with the specification. `GRIP_DENY_POLICY`, the implementation-defined code, **must** carry a namespaced machine-readable subcode (`X_{VENDOR}_*`) plus prose, and conformance reporting surfaces each deployment's POLICY fraction. Registries (verbs, classes, boundaries, reasons, budget axes) accept extensions by public proposal.

4.5 Budgets

The cost object has four registered axes — `usd_micro`, `tokens_in`, `tokens_out`, `ms`. The monetary axis meters the monetary consequence recorded for an act: inference fees, service charges, and transfer values alike (a $42 refund is an act with `usd_micro: 42000000`). One axis meters two economically different things — the fee paid to do work and the value moved by the work itself. GRIP/1.0 keeps them in one axis for simplicity and reserves `transfer_value_usd_micro` in the registry for deployments that must bound them separately; the choice is a known open question rather than a settled one. Admission precedence is deterministic: an act that breaches a present budget axis is refused with `GRIP_DENY_BUDGET` **before** any Stop is raised, so a human is never asked to approve an act the contract has already denied. Axes are extensible by registry; **an unknown axis present in a governing budget is criticality-bearing by default** — a runtime that cannot meter it must refuse acts under that Grant (`GRIP_DENY_UNKNOWN_CRIT`), because silently ignoring a bound the issuer imposed is a widening, not an

extension. Budget arithmetic is defined on the chain: an act's cost charges every Grant on its authority path and the Mandate; Grant issuance **reserves** the child's budget against the parent's remaining axes, so sibling Grants and delegated sub-chains cannot jointly exceed a parent by racing (admission at GRIP-1+ must be serialized against the chain head); the delegator charges the full reserved slice at issuance and reconciles on the delegate's report. At GRIP-1 and above a runtime must deny — not merely record — an act whose estimated cost exceeds a remaining axis, and outcome actuals supersede estimates (§4.3.3).

4.6 Chain verification

The verifier released with this paper implements, and the specification defines, the audit algorithm any third party can run from the trust bundle plus an object set:

1. **Signatures**: every object verifies against its issuer's published key, within the key's validity window; at GRIP-2+, `act.actor_sig` verifies against the Grant's actor key.
2. **Chain integrity**: `prev`-links unbroken per Mandate; `parents` resolve; referenced digests match supplied content.
3. **Authority**: every Receipt's Grant chain reaches the Mandate; the root Grant's `iss` is in the Mandate's `issuers`; the dominance laws hold at every hop (machine-checked against `scope` where present; the root hop is reported issuer-attested where not); every act falls within its Grant's `allow` under the severity-dominant matching rule, with `grant.iat ≤ receipt.iat < grant.exp`.
4. **Budget**: recomputed running totals — outcome actuals superseding paired estimates, reservations charged at issuance — never exceed any present axis of any ancestor at any chain prefix.
5. **Stops**: every allowed settle-class act references an approved, unexpired, unconsumed Stop whose `act_digest` equals the act's `input_digest`, whose approval is attributable to the named approver under the declared mode, and which no earlier allowed act already consumed (or whose multi-use envelope covers this act).
6. **Settlement and closure**: every acceptance criterion addressed; independent-marked criteria carry validator signatures satisfying the reported predicate; `totals` match rule 4's recomputation; no Receipt's `iat` postdates the Mandate's terminal Settlement (`GRIP_DENY_SETTLED` governs the gate side).
7. **Context**: every model-driven allowed act's Receipt references a Context Envelope; items carry trust classes; `used_tokens ≤ budget_tokens`. (Rendering constraints are conformance-clause territory and are not claimed as chain-verified.)
8. **Anchoring and time**: the chain extends the latest head attestation known to the verifier (below), and no object's `iat` precedes its parents'.

The output is a per-rule verdict, not a scalar score. A branch a conformant gate denied for an unknown critical extension receives the distinct verdict **`unevaluable-critical`** — correctly-refused authority is not chain failure.

Anchoring. `prev`-links authenticate ordering *within the disclosed set*; they cannot detect a withheld suffix or a parallel fork. GRIP/1.0 therefore requires, at GRIP-2 and above, periodic signed **head attestations** — `{mandate, head, receipt_count, running totals}` — delivered to the Principal's designated surface, and recommends counterparty co-signatures on settle-class outcome Receipts (the Sello pattern [13]). A chain that cannot be shown to extend its latest attestation is reported **incomplete**, not clean. Truncation, equivocation, and issuer-asserted clocks remain named residual risks (§6); external transparency-log anchoring is a reserved extension.

4.7 Conformance levels

Level	Name	Normative requirement	What it honestly is
GRIP-0	Witness	Emit signed Receipts on one chain; publish a trust bundle. Mandate-less witness chains permitted (the named exception to P5); `parents` absent	Self-reported, verifiable audit records
GRIP-1	Accountant	Mandates with `issuers` and budgets; over-budget acts denied; actuals reconciled; serialized admission	A spend ceiling with a paper trail
GRIP-2	Warden	Grants with dominance enforced at issuance and act time; actor signatures; act-bound Stops on settle-class acts; head attestations	Enforcement outside the agent
GRIP-3	Sovereign	Context Envelopes with trust classes and taint; Settlement with signed independent validation; all eight verification rules pass	The complete contract

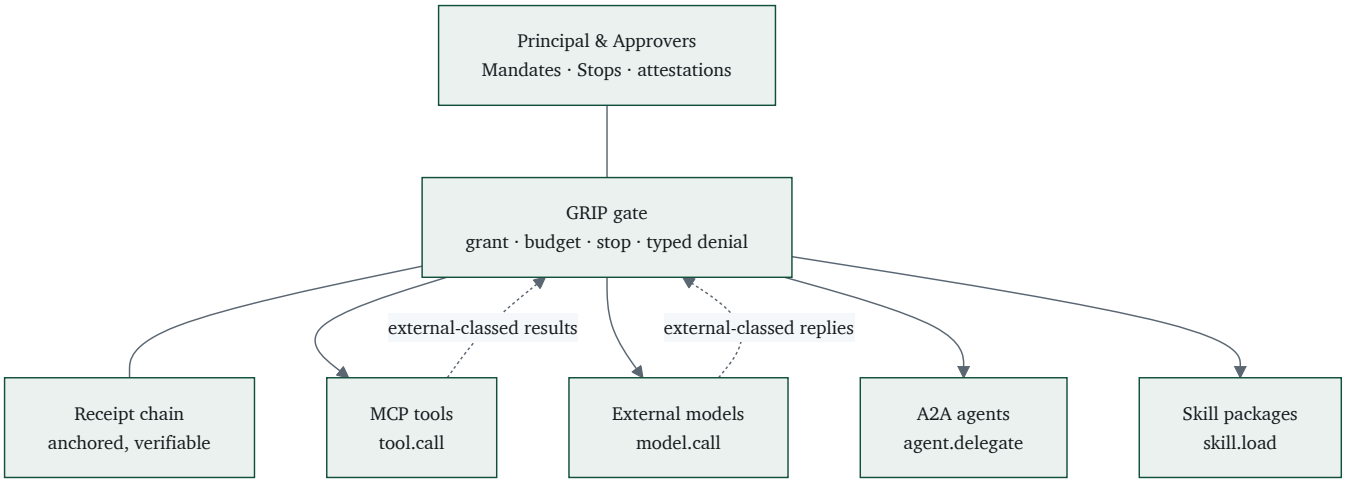
Levels are cumulative; the conformance suite released with this paper tests each, and its negative cases — forged Grant, replayed Receipt, over-budget act, fabricated Stop approval, reused Stop, tampered chain — pass only by *refusing* correctly. Implementations advertise their level; A2A Agent Cards can carry it via the v1.0.1 extension mechanism (§5.2).

4.8 Versioning and evolution

Non-critical GRIP/1.x additions must be backward-compatible under must-ignore. Additions that change authorization semantics — the reserved revocation and attestation extensions among them — **must** be `crit`-marked and will be refused by older consumers *by design*: that refusal is the interoperability contract, not a failure of it. Registries evolve by public proposal; GRIP/2.0 is reserved for changes that break verification.

5. Composition with the existing stack

Figure 3 — one gate, five families:



5.1 MCP, in both directions

GRIP over MCP (downward). A runtime acts as MCP client. Grant entries scope tools by resource pattern and consequence class; observe-class tools flow, settle-class tools require Stops, egress-class tools are visible as such in the chain. Tool *results* re-enter as `external` context items; tool *descriptions* render as the server's words with no instruction privilege. MCP's authorization governs the client–server connection; GRIP adds what MCP deliberately leaves out — who authorized this particular act, under what bounds, approved by whom.

GRIP under MCP (upward). A GRIP runtime is itself exposable as an MCP server, so existing MCP-speaking agents dock into governance through a protocol they already implement: their tool calls arrive as acts, their world-changing calls suspend into Stops, their sessions leave chains. This is the lowest-friction adoption path: governance without agent modification.

5.2 A2A

Delegating to an external agent is `agent.delegate`: an attenuated, expiring, budget-reserved Grant out; a sub-chain (or at minimum a settlement-style outcome report) back, reconciled against the reservation so a delegate's self-reporting can understate its own spend but never overspend the delegator's books. Agent Cards advertise a GRIP conformance level through A2A's extension mechanism; the delegator's chain records the attested level, so an audit distinguishes "verified sub-chain" from "self-reported outcome." Richer federation — shared settlement, verifiable partial disclosure between estates — is out of GRIP/1.0 (§10).

5.3 Skills and models

Both are §4.4's normative territory; composition adds only this: skill formats are consumable if content-addressable (existing open skill-package conventions qualify), and organizations' model policy ("customer data never leaves our cloud"; "reasoning spend ≤ $x per task") compiles to Grant fields (`boundary`, budget) whose enforcement is chain-checkable through `outcome.resolved` — with the tool-wrapped-model evasion named in §6 rather than papered over.

5.4 Telemetry

Receipts map onto OpenTelemetry GenAI spans [29] (act ↔ span, cost ↔ attributes, chain ↔ trace); the mapping table ships with the artifacts. Existing observability pipelines ingest GRIP without new plumbing, and a span-

emitting agent is one adapter from GRIP-0.

5.5 Payments

AP2-style payment mandates [5] compose as an extension to settle-class acts: a purchase's Stop approval and Settlement evidence align with the payment protocol's own signed mandate chain. GRIP defines no payment rails and needs none.

5.6 Transport binding (informative for 1.0)

Interoperability needs a wire, so GRIP/1.0 states one as SHOULD-level convention: the trust bundle at `/.well-known/grip`; chain retrieval as HTTPS GET of JSON object sets by Mandate id from the origin the trust bundle names; objects conveyed in the MCP binding as tool metadata/content blocks and in the A2A binding as task artifacts. Implementations may substitute transports; the objects, not the pipes, are the protocol.

6. Security analysis

Threat model. In scope: a compromised or manipulated model (prompt injection, adversarial tool output, malicious skill content); a malicious tool server; an over-permissive or careless operator; an honest-but-curious verifier; replay, forgery, and reuse attacks on the objects; approval-surface spoofing; delegate over-reporting. Out of scope for v1: a compromised runtime gate (receipts detect tampering with disclosed objects; anchoring bounds — but does not eliminate — suppression; issuer separation narrows blast radius); root key theft; denial of service.

Properties and arguments.

- **Authority provenance and designated issuance.** Every executed act chains by signature and hash to a human-signed Mandate *through an issuer that Mandate names* (rules 3): trust-bundle membership alone cannot mint authority, and at GRIP-2+ actorship itself is signed, so a leaked Grant is unusable without the actor's key.
- **Attenuation-only delegation.** The dominance laws are machine-checked Grant-to-Grant, and at the root hop whenever the Mandate carries a `scope`; verifiers disclose which. Escalation below the first Grant is a verification failure; the first Grant is an accountable, designated, signed decision. Skills cannot widen authority structurally.
- **Act-bound, single-use, attributable approval.** A Stop names its act digest and approver; signature-mode approval cannot be fabricated by the runtime; consumption rules make approve-one-execute-many a rule-5 failure; the approval-surface clause makes bait-and-switch a conformance violation rather than an undetectable rendering trick. Token mode's weaker trust is stated, not hidden.
- **TTL containment.** All authority expires; a leaked credential's damage window is its remaining lifetime, budget, and (at GRIP-2+) the actor-signature requirement. No revocation in v1 is a stated tradeoff with a reserved, `crit`-safe extension.
- **Bounded, attributable injection — write-side.** All five act families return through one classed, framed, budgeted door; what a hijacked step can *change* is limited by Grant, budget, and Stops, and the poisoned item is identifiable by envelope digest. **Read-side disclosure is a named residual, not a claim**: GRIP does not bound what an injected agent may read and exfiltrate *within its authorized envelope* through observe/egress-class acts — the `egress` class exists to make that surface visible and gateable, and argument-level constraint predicates (the Progent [11] design point) are a reserved extension.

- **Chain verifiability with stated limits.** The eight rules run from the trust bundle alone; head attestations and settle-class co-signatures bound suppression and equivocation; digests bind integrity, salted commitments handle confidentiality; timestamps are issuer-asserted within rule-8 ordering constraints.

Residual risks, named. Consequence-class misclassification by a careless operator removes Stop protection undetectably in-chain (verifiers therefore report settle-free resources). Tool-wrapped models evade `model.*` accounting up to Grant-author diligence. Token-mode approvals trust the runtime's discard promise and the delivery path. Identity-inequality independence does not resist sock-puppet validators under one principal (stronger predicates are profile extensions; settlements report their predicate). Chain suppression between attestations, and clock dishonesty within rule-8's ordering, remain possible windows. GRIP-0/1 receipts are self-reported by definition. And no protocol governs the authority an operator grants wholesale: GRIP makes broad authority visible, designated, and expiring — not impossible.

7. GRIP and Agentic Language Models

A governance protocol fixes what a model *sees* and *emits* at every step; GRIP was designed so this interface is also a training interface — a property we have not found in the protocols surveyed in §2, though every mechanism it composes has prior art. Three design choices carry it. **Schemas compile to grammars**: the six objects and the act registry are JSON Schemas released with this paper; constrained-decoding engines [24] turn them into grammars, so syntactic validity is enforceable at the decoder and the model's learning burden is judgment (with known engine-coverage caveats [25]). **Denials are labeled negatives**: every typed denial is a machine-labeled example — this act, in this context, fails for this reason — produced as exhaust by every conformant runtime, for any vendor. **Settlements are outcome labels**: evidence-gated verdicts label whole trajectories without annotation.

We call a model native to this interface an **Agentic Language Model (ALM)** — the acronym has prior uses (Augmented Language Models [35]; a task-decomposition framework [36]); the class defined here and in the companion paper is distinct: a model defined by the governance contract it operates under. The composition of §5 is what makes a *small* native model plausible in the seat. Its decisions are closed — which tool, which model class, delegate or not, stop or proceed; its authority is external; and its economy (solve locally or buy frontier cognition via `model.call`) is trained by real budget pressure. Trained-orchestrator results [19, 20, 21] establish the capability side; the governance-native interface is what has been missing. Because the schemas, registries, and denial semantics are open, the flywheel is not proprietary: any organization adopting GRIP can train its own contract-native models, and any GRIP-native model is portable across conformant runtimes. A companion paper develops Fennec, one such model; GRIP does not depend on it.

8. Reference implementation and artifacts

Released with this paper: **(a)** the JSON Schemas, registries, and conformance suite; **(b)** `grip-verify`, the standalone verifier implementing §4.6, whose canonical test vector is the worked chain of Appendix C.

Two further artifacts are **specified but not yet released**, and this paper marks them as such rather than let the reader assume otherwise: **(c)** SDK middleware (TypeScript, Python) intended to wrap existing agent loops — framework agents and MCP-speaking clients — to GRIP-0 without restructuring; and **(d)** Foxora Runtime, the governed agent engine this specification was drawn from, which implements the GRIP-3 semantic core — gate-enforced grants, transactional budget accounting, receipted context compilation, and evidence-gated settlement

— and serves as the protocol's living testbed. Its source release is a submission gate, not a completed one: at this draft the engine is not published, so Appendix D's correspondence table is checkable against this specification but not yet against public code, and a reader should treat it as an author attestation until it is. Its implementation status is tracked in a gap ledger rather than asserted, and **Appendix D maps every specification object to the runtime type that implements it, with file and line references**, so the claim in this paragraph is checkable rather than taken on trust: at this draft, the stop-issuance surface is implemented but not yet mounted in the shipped engine, and the full governed chain has not yet been driven as a single run — both are submission gates (header), not footnotes. Approval surfaces (desktop or web applications) are implementation territory; none is required by, or privileged in, the protocol, and nothing in §4–§6 depends on this engine.

9. Evaluation

A protocol debut is evaluated on coverage, adoption cost, and security argument — not on benchmarks it has no deployments to run. We are explicit about which claims are scored self-assessment (rubric released for re-grading), which are measurements pending released artifacts, and which are design arguments.

Coverage (author-scored; every cell cites its section). Two instruments, kept apart. Table 1 (§2) is our own execution-governance rubric and grades seven protocols against the questions of §1. Separately, against the community-governance taxonomy of [6]: GRIP addresses **human escalation** (Stops, §4.3.4) and **audit/replay** (the receipt chain and §4.6 verification) and does **not** address **membership**, **deliberation**, **voting** or **dissent preservation** — those govern how a collective decides, which a per-execution contract deliberately does not model. We report two of six, and claim no more. Against the six-dimension taxonomy of [6]: authority and delegation — §4.3.1–§4.3.2 (designated issuers, dominance laws); accountability — §4.3.3, §4.6 rules 1–3 (signed receipts, actor signatures, attribution); resource bounds — §4.5, rule 4; human oversight — §4.3.4, rule 5; evidence and audit — §4.6, anchoring. The sixth dimension, community governance and dissent aggregation, is deliberately out of scope for a per-execution contract; [6] finds it universally absent and GRIP does not change that. Against the five auditability dimensions of [7]: action recoverability (Receipts, outcome pairing), lifecycle coverage (Mandate→Settlement), policy checkability (typed reasons, dominance laws), responsibility attribution (signatures, designated issuance, validator signatures), evidence integrity (content addressing, chains, attestations). These scorings are ours; the machine-readable rubric ships with the artifacts, and §10 stands on the point that self-assessment is where every protocol debut starts and external re-grading is the exit.

Adoption cost (to be measured on the released SDK). The witness tests of the conformance suite are designed to pass against an unmodified agent loop plus middleware; lines-of-integration and time-to-first-verified-receipt for the SDK examples will be reported in the artifact repository, and P1's "afternoon" is the design target those numbers test.

Security argument. §6, with the negative conformance cases (forged Grant, replayed Receipt, over-budget act, fabricated Stop approval, reused Stop, tampered chain) encoded in the suite released with this paper — an implementation reaches GRIP-2+ only by refusing correctly. The worked chain of Appendix C is the verifier's canonical input; its eight-rule verdict is reproducible by any reader from the published objects.

What this paper does not evaluate. Runtime overhead microbenchmarks (to be published with the reference engine — [7]'s 8.3 ms and [8]'s sub-3 ms figures are context from other systems, not comparators); multi-runtime federation; and model-side results, which belong to the companion ALM paper.

10. Limitations and future work

GRIP/1.0 has no external implementers yet; the specification, conformance suite, and verifier are the invitation, and the coverage scoring is self-assessed until others re-grade it. Revocation is deliberately absent (TTL containment instead) and is the first reserved, `crit`-marked extension; runtime attestation and external transparency-log anchoring are the second and third. Argument-level constraint predicates on `allow` entries — bounding *what* may pass through an authorized act, the read-side exfiltration residual of §6 — are reserved with the Progent line [11] as prior art. The independence predicate is minimal (identity inequality) with stronger predicates left to profiles. Cross-runtime federation with verifiable partial disclosure is designed territory but out of v1. Formal verification of the state machines and dominance laws, and a CBOR/COSE profile, are future work. The model-side program — training contract-native policies against this protocol and measuring whether governance-native beats governance-wrapped — is the companion paper's subject.

11. Conclusion

Agent capability got its protocols; agent governance did not, and the world has been compensating with wrappers, one private vocabulary at a time. GRIP's wager is that governance, like transport and access before it, is a protocol problem: six signed objects, one anchored chain, five act families, four levels — small enough to adopt in an afternoon, strict enough to prove who authorized what, at what cost, with whose approval, to any verifier. The specification, schemas, verifier, conformance suite, and reference engine are released with this paper. The protocol is the invitation.

Artifact availability

Released with this paper, at `grip.foxora.dev`: the specification text, seven JSON Schemas, five registries, `grip-verify`, and the conformance suite with its seven signed test vectors — schemas and registries under CC-BY-4.0, code under Apache-2.0. The canonical chain of Appendix C verifies clean under all eight rules; the six negative vectors each fail exactly the rule they target, and `conformance/run.py` asserts that they fail no others.

Not yet released, and named here so the distinction is not left to inference: the reference runtime's source, the SDK middleware, and an end-to-end chain driven as a single live run. Appendix D maps this specification to that implementation object by object, with file and line references; until the source is public, that table is an attestation rather than something a reader can check.

References

[1] Model Context Protocol Specification, rev. 2026-07-28. modelcontextprotocol.io. [2] Agent2Agent (A2A) Protocol Specification, v1.0.1, released 28 May 2026, tag v1.0.1, commit 3303592. Negotiated wire version remains 1.0; patch numbers are excluded from negotiation (§3.6). [3] AGNTCY Collective, Agent Connect Protocol (ACP) Specification, v0.2.3; github.com/agntcy/acp-spec — archived read-only 11 April 2026. [4] T. Meunier & S. Major, "HTTP Message Signatures for Automated Traffic", Internet-Draft draft-meunier-webbotauth-httpsig-protocol-02, 18 August 2026. Work in progress; individual submission, not adopted. [5] Agent Payments

Protocol (AP2), v0.2.0, 28 April 2026, Apache-2.0; contributed to the FIDO Alliance 28 April 2026, where standardisation continues. ap2-protocol.org. [6] Kang & Diponegoro. “Governance Gaps in Agent Interoperability Protocols: What MCP, A2A, and ACP Cannot Express.” arXiv:2606.31498, 2026. [7] Nian et al. “Auditable Agents.” arXiv:2604.05485, 2026. [8] Prakash. “AIP: Agent Identity Protocol for Verifiable Delegation Across MCP and A2A.” arXiv:2603.24775, 2026. [9] South, Marro, Hardjono, Mahari, Whitney, Greenwood, Chan, Pentland. “Authenticated Delegation and Authorized AI Agents.” arXiv:2501.09674, 2025. [10] Debenedetti et al. “Defeating Prompt Injections by Design” (CaMeL). arXiv:2503.18813, 2025. [11] “Progent: Securing AI Agents with Privilege Control.” arXiv:2504.11703, 2025. [12] “Proof-Carrying Agent Actions.” arXiv:2606.04104, 2026. [13] Figuera. “Notarized Agents (Sello).” arXiv:2606.04193, 2026. [14] “Deontic Policies for Runtime Governance of Agentic AI Systems.” arXiv:2606.19464, 2026. [15] Yao et al. “τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains.” arXiv:2406.12045, 2024. [16] Barres, Dong, Ray, Si, Narasimhan. “τ^2-bench.” arXiv:2506.07982, 2025. [17] Andon Labs. “Vending-Bench.” arXiv:2502.15840, 2025. [18] Belcak et al. “Small Language Models are the Future of Agentic AI.” arXiv:2506.02153, 2025. [19] Su, Diao, Lu et al. “ToolOrchestra: Elevating Intelligence via Efficient Model and Tool Orchestration.” arXiv:2511.21689, 2025. (Released model: nvidia/Nemotron-Orchestrator-8B.) [20] Yuan et al. “Small Model as Master Orchestrator (ParaManager).” arXiv:2604.17009, 2026. [21] Mangannavar, Coalson, Dugar, Tadepalli. “Training the Orchestrator: A Supervised Approach to End-to-End PDDL Planning with LLM Agents” (HALO). arXiv:2606.21740, 2026. [22] Ross, Mahabaleshwarkar, Suhara. “When2Call: When (not) to Call Tools.” arXiv:2504.18851, NAACL 2025. [23] “BAGEN: Are LLM Agents Budget-Aware?” arXiv:2606.00198, 2026. [24] “XGrammar.” arXiv:2411.15100, 2024; XGrammar-2, arXiv:2601.04426, 2026. [25] “JSONSchemaBench.” arXiv:2501.10868, 2025. [26] Regulation (EU) 2024/1689 (AI Act), Article 12, as amended by Regulation (EU) 2026/1744 (AI Omnibus), in force 27 July 2026: Chapter III Sections 1–3 apply from 2 December 2027 (Annex III) and 2 August 2028 (Annex I). [27] RFC 8785: JSON Canonicalization Scheme (JCS). [28] RFC 7515: JSON Web Signature; RFC 8037: EdDSA in JOSE. [29] OpenTelemetry Semantic Conventions for Generative AI. opentelemetry.io. [30] SPIFFE: Secure Production Identity Framework for Everyone. spiffe.io. [31] Birgisson, Politz, Erlingsson, Taly, Vrable, Lentczner. “Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud.” NDSS 2014. [32] Biscuit authorization tokens. biscuitsec.org. [33] RFC 9396: OAuth 2.0 Rich Authorization Requests; RFC 8693: OAuth 2.0 Token Exchange. [34] RFC 7797: JSON Web Signature (JWS) Unencoded Payload Option. [35] Mialon et al. “Augmented Language Models: a Survey.” arXiv:2302.07842, 2023. [37] Zhu, Wang, Wang, Li & Li. “OpenPort Protocol: A Security Governance Specification for AI Agent Tool Access.” arXiv:2602.20196, 22 February 2026. [38] Zhu & Wang. “Intent-Governed Tool Authorization for AI Agents.” arXiv:2606.22916v3, 29 July 2026. [39] Wu. “SessionBound: Turning Enterprise Task Approval into Budgeted Database Sessions.” arXiv:2607.00751, 2026. [40] Liu, Wu et al. “Foundation Protocol: A Coordination Layer for Agentic Society.” arXiv:2605.23218, 22 May 2026. Reference runtime: github.com/FoundationAgents/foundation-protocol (MIT).

[36] Maurya & Maurya. “Agentic Language Model (ALM): A Task-Centric Framework for Autonomous AI Systems.” IRE Journals 9(10), 2026.

Appendix A — Denial-reason registry (initial)

Code	Meaning
`GRIP_DENY_SCOPE`	act outside the Grant's `allow` under the dominant-class matching rule
`GRIP_DENY_BUDGET`	a present budget axis of any ancestor would be exceeded
`GRIP_DENY_EXPIRED`	Grant, Mandate, or the referenced Stop expired at act time
`GRIP_DENY_STOP_REQUIRED`	settle-class act with no approved, unconsumed, act-matching Stop
`GRIP_DENY_STOP_REJECTED`	the referenced Stop was rejected
`GRIP_DENY_SETTLED`	the Mandate has a terminal Settlement
`GRIP_DENY_RATE`	implementation concurrency/fairness bound (incl. stop.request discipline)
`GRIP_DENY_UNKNOWN_CRIT`	object carries an unimplemented critical extension or unmeterable budget axis; `decision.crit_object` names it
`GRIP_DENY_POLICY`	implementation-defined; namespaced subcode + prose required

Appendix B — Conformance checklist (summary)

GRIP-0: envelope and signature validity; chain integrity; trust bundle at `/.well-known/grip`; witness tests pass (mandate-less permitted). GRIP-1: Mandates with `issuers` and budgets; reservation and serialization; over-budget denial; actual-cost reconciliation. GRIP-2: dominance laws at issuance and act time; actor signatures; act-bound single-use Stops with attributable approval; head attestations; negative tests refused. GRIP-3: Context Envelopes with trust classes and taint propagation; Settlement with signed independent validation; all eight verification rules pass end-to-end on the Appendix C chain shape. Machine-readable checklist and test vectors ship in the artifact repository.

Appendix C — Worked chain (the refund scenario, end to end)

The canonical `grip-verify` test vector, published in full at `/grip/vectors/refund-happy-path/`: dana's Mandate (§4.3.1) → coordinator's Grant (§4.3.2) → Context Envelope (§4.3.6) → a governed `model.call` pair whose outcome Receipt records the resolved provider and matched boundary (§4.4) → denied refund attempt (§4.3.3, `GRIP_DENY_STOP_REQUIRED`) → act-bound Stop (§4.3.4) → dana's signature-mode approval:

```
{ "grip": "1.0-draft.2", "type": "receipt",
  "id": "sha256:500e…", "parents": ["sha256:3de8…", "sha256:2f98…"],
  "prev": "sha256:f64b…",
  "iss": "https://runtime.acme.com/issuers/gate#key1",
  "actor": "https://id.acme.com/people/dana#key1",
  "act": { "verb": "stop.approve", "resource": "sha256:2f98…" },
  "approval": "…detached JWS by dana#key1 over (stop.id ‖ act_digest)…",
  "decision": { "result": "allowed" },
  "cost": { "usd_micro": 0, "tokens_in": 0, "tokens_out": 0, "ms": 2 },
  "iat": 1787000900, "sig": "…" }
```

→ the allowed settle-class refund (gate Receipt referencing the consumed Stop; paired outcome Receipt with `outcome: {status: "ok", effects: ["crm:ticket/8812#refund"], resolved: null}`, actual `cost.usd_micro: 42000000`) → the Settlement (§4.3.5) addressing both criteria, `refund-ledger` validated and signed by an identity that took no act in the chain.

Every digest printed above is the object's real content-address, reproducible by regenerating the vectors: the keys are seeded, so the chain is byte-identical on any machine. Eleven objects; `grip-verify` returns **clean**, all eight rules `pass`, recomputed totals `{usd_micro: 42480000, tokens_in: 413000, tokens_out: 21100, ms: 2399990}`.

The suite also ships six negative vectors, each a *fully consistent chain differing from the happy path in exactly one way* — a vector that fails several rules at once proves nothing about any of them:

Vector	Defect	Must fail
`negative-stop-replay`	a second act references the consumed Stop	rule 5
`negative-fabricated-approval`	the approval is not the approver's signature	rule 5
`negative-forged-grant`	root Grant from an issuer the Mandate does not list	rule 3
`negative-over-budget`	outcome cost breaches the Grant ceiling, **with the Settlement totals restated to match**	rule 4
`negative-post-settlement`	a Receipt dated after the terminal Settlement	rule 6
`negative-tampered-chain`	the Mandate edited after signing	rules 1, 2

The over-budget vector is the sharpest of the six: an adversary who also rewrites the Settlement produces a self-consistent document, and rule 4 catches it anyway, because the ceiling is recomputed from the Receipts rather than read from the summary. `conformance/run.py` asserts each vector fails its rules **and no others**; at this draft it reports 7/7.

Appendix D — Specification-to-implementation correspondence

The reference runtime predates this specification and uses its own vocabulary: it speaks protobuf over a transactional store, not JSON over HTTP. The governance semantics are nonetheless the same objects, and the table below is the map a reader needs to check §8's claim for themselves. Every row cites a file and line in the released implementation.

Spec object	Runtime type	Location	Correspondence
Mandate	`IntentContract`	`intent.proto:33`	**Exact** — desired state, constraints, acceptance criteria, four-axis budget, deadline, risk, six-state machine
Grant	`CapabilityLease`	`identity.proto:27`	**Exact** — principal, subject, resource + actions, budget, validity window, non-delegable, Ed25519 over canonical claims
Receipt	`CognitiveEvent` ledger	`ledger.proto:9`	**Composite** — the spec's per-act Receipt corresponds to a sealed, parent-linked ledger event plus its command receipt and budget charge; the runtime splits what the spec unifies
Stop	`WaitpointRecord`	`waitpoint.proto:84`	**Partial — spec ahead** — the runtime binds a waitpoint to an intent, reason and evidence requirements, and stores only a token digest; the specification's act-digest binding and single-use consumption (§4.3.4) are hardening this paper introduces and the engine does not yet implement
Settlement	`SettlementReceipt`	`evidence.proto:73`	**Exact** — per-criterion validation, evidence references, residual risk, kernel signature over a domain-separated payload
Context Envelope	`CognitiveCapsule`	`cognition.proto:77`	**Exact in substance** — purpose, role, provenance-carrying references, token budget and usage, SHA-256 content digest, paired selection receipt

Two honest notes. First, the runtime signs protobuf under its own domain-separated Ed25519 scheme rather than JOSE over JCS; a conformant GRIP/1.0 export is an encoding adapter over the same objects, not a re-architecture. Second, where the table reads *spec ahead*, the specification is deliberately stricter than the implementation — the

act-bound, single-use Stop closes an amplification and bait-and-switch weakness identified during review of this paper. Those hardenings are tracked in the implementation's public gap ledger.