# GhostLedger

## API Specification

*Complete REST API reference for the Internet Reality Engine*

Version 1.0 — February 2026

CONFIDENTIAL

# Table of Contents



*"Substance in words — every endpoint exists because a real action demands it."*

# 1. Overview & Base URL

The GhostLedger API exposes every capability of the Internet Reality Engine as a RESTful service. All endpoints return JSON, accept JSON request bodies (Content-Type: application/json), and follow consistent error handling. Every mutating operation is logged to the audit trail and eligible for on-chain anchoring on Solana.

| Property | Value |
|---|---|
| **Base URL** | `https://api.ghostledger.io/v1` |
| **Protocol** | `HTTPS (TLS 1.3)` |
| **Format** | `JSON request and response bodies` |
| **Auth** | `Bearer token (API key or JWT)` |
| **Versioning** | `URL-path versioned (/v1, /v2, ...)` |
| **Pagination** | `Cursor-based (limit + cursor params)` |
| **Idempotency** | `X-Idempotency-Key header on all POSTs` |
| **Rate Limits** | `Per-key, returned in X-RateLimit-* headers` |
| **Chain Layer** | `Solana (SHA-256 anchoring)` |

Required Headers:

```
Authorization: Bearer <api_key>
Content-Type: application/json
X-Request-ID: <uuid>           # optional, for tracing
X-Idempotency-Key: <string>    # optional, prevents duplicates
```

## Environments:

| Environment | Base URL | Purpose |
|---|---|---|
| **Production** | `https://api.ghostledger.io/v1` | Live system |

| | | |
|---|---|---|
| **Staging** | `https://staging-api.ghostledger.io/v1` | Pre-release testing |
| **Local** | `http://localhost:8080/v1` | Developer sandbox |

## Design Principles:

- Consent-first — No action on a victim without explicit consent flowing through the ConsentUnlock gate.
- Three-signature governance — Every significant state change passes through Verifier + Policy + Risk. Any signer can veto.
- Immutable audit — Every API call that mutates state is logged. Critical records are anchored on Solana.
- Deterministic policy — The 7 Laws are enforced by rules evaluated before any state transition. No exceptions.
- Privacy by default — Victim profiles are pseudonymized. No PII leaves the system through any endpoint.

# 2. Authentication

GhostLedger uses API keys for service-to-service authentication and JWT tokens for user-facing sessions. Every key is scoped to a role with least-privilege permissions.

API Key Scopes:

| Scope | Access Level | Use Case |
|---|---|---|
| read:noise | Read-only NIE data | Dashboards, monitoring |
| write:noise | Submit noise events | Ingestion agents |
| read:cases | View claims & cases | Case managers, HRN members |
| write:cases | File and update claims | Case management agents |
| admin:governance | Policy rules, decisions | Governance operators |
| read:litmus | View evaluations | Public LITMUS scores |
| write:litmus | Trigger evaluations | LITMUS evaluation agents |
| admin:agents | Agent lifecycle | System administrators |
| write:hrn | HRN engagements | HRN coordination agents |
| admin:chain | Chain anchoring | Anchor service |

JWT Token Format:

```
POST /v1/auth/token
{
  "api_key": "gl_live_...",
  "scopes": ["read:noise", "read:cases"]
}

Response:
{
  "token": "eyJhbGciOi...",
  "expires_in": 3600,
  "scopes": ["read:noise", "read:cases"]
}
```

# 3. Error Model

Every error response follows a consistent structure. The error code is machine-readable; the message is human-readable. Details provide field-level validation errors when applicable.

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Request body failed validation",
    "request_id": "req_abc123",
    "details": [
      {"field": "severity", "issue": "Must be one of: low, medium, high, critical"}
    ]
  }
}
```

Standard Error Codes:

| HTTP | Code | Meaning |
|---|---|---|
| 400 | VALIDATION_ERROR | Invalid request body or parameters |
| 401 | UNAUTHORIZED | Missing or invalid API key / token |
| 403 | FORBIDDEN | Key lacks required scope |
| 404 | NOT_FOUND | Resource does not exist |
| 409 | CONFLICT | State conflict (e.g., claim already resolved) |
| 422 | POLICY_VIOLATION | Action blocked by governance rule |
| 429 | RATE_LIMITED | Too many requests |
| 500 | INTERNAL_ERROR | Server error (auto-reported) |
| 503 | SERVICE_UNAVAILABLE | Dependent service is down |

# 4. Noise Intelligence API

The Noise Intelligence Engine ingests raw signals from across the internet, classifies them as Chaos, Rumor, or Hype, and clusters them into Storms. These endpoints expose the full lifecycle.

## 4.1 Noise Events

`POST` `/noise/events` Submit a noise event

Request Body:

```
{
  "noise_type": "rumor",
  "platform": "twitter",
  "source_url": "https://twitter.com/...",
  "content_snippet": "Major exchange reportedly insolvent...",
  "severity": "high",
  "confidence": 0.82,
  "entities": ["ExchangeX"],
  "tags": ["insolvency", "exchange"]
}
```

Response:

```
201 Created
{
  "event_id": "evt_a1b2c3d4",
  "noise_type": "rumor",
  "severity": "high",
  "storm_id": null,
  "created_at": "2026-02-06T14:30:00Z"
}
```

`GET` `/noise/events` List noise events with filters

Path / Query Parameters:

- `noise_type` — Filter by chaos | rumor | hype

- `platform` — Filter by platform name
- `severity` — Filter by severity level
- `entity` — Filter by entity name
- `since` — ISO 8601 timestamp lower bound
- `until` — ISO 8601 timestamp upper bound
- `limit` — Results per page (default 50, max 200)
- `cursor` — Pagination cursor

Response:

```
200 OK
{
  "events": [ ... ],
  "cursor": "cur_xyz789",
  "total": 1247
}
```

GET `/noise/events/{event_id}` Get a single event

Path / Query Parameters:

- `event_id` — UUID of the noise event

Response:

```
200 OK
{
  "event_id": "evt_a1b2c3d4",
  "noise_type": "rumor",
  "platform": "twitter",
  "content_snippet": "...",
  "severity": "high",
  "confidence": 0.82,
  "entities": ["ExchangeX"],
  "storm_id": "stm_e5f6g7",
  "evidence": [ ... ],
  "created_at": "2026-02-06T14:30:00Z"
}
```

# 5. Storm & Front API

Storms are clusters of related noise events that form coherent narratives. When multiple Storms converge around shared entities and timing, they form a Front — indicating systemic patterns that warrant deep investigation.

## 5.1 Storms

`GET` `/storms` List active storms

Path / Query Parameters:

- `status` — forming | active | escalated | resolved
- `severity` — Filter by severity
- `entity` — Filter by entity
- `limit` — Results per page (default 25)

Response:

```
200 OK
{
  "storms": [
    {
      "storm_id": "stm_e5f6g7",
      "title": "ExchangeX Insolvency Rumor",
      "status": "active",
      "severity": "high",
      "score": {"velocity": 12.5, "composite": 6.82},
      "event_count": 47,
      "platforms": ["twitter", "reddit", "discord"]
    }
  ]
}
```

`GET` `/storms/{storm_id}` Get storm detail

Returns full storm object including all linked event IDs, score breakdown, evidence refs, and front assignment.

Path / Query Parameters:

- `storm_id` — UUID of the storm

`GET` `/storms/{storm_id}/events` List events in a storm

Path / Query Parameters:

- `storm_id` — UUID of the storm
- `limit` — Results per page

`POST` `/storms/{storm_id}/escalate` Escalate a storm

Requires write:noise scope. Triggers a governance Decision gate — the storm is not escalated until all three signatures approve.

Request Body:

```
{
  "reason": "Cross-platform acceleration detected",
  "target_severity": "critical"
}
```

## 5.2 Fronts

`GET` `/fronts` List active fronts

Path / Query Parameters:

- `status` — detected | monitoring | active | escalated
- `severity` — Filter by severity

`GET` `/fronts/{front_id}` Get front detail

Path / Query Parameters:

- `front_id` — UUID of the front

Response:

```
200 OK
{
  "front_id": "frt_h8i9j0",
  "title": "DeFi Lending Protocol Collapse Cluster",
  "status": "active",
  "severity": "critical",
  "score": {"storm_count": 4, "composite": 8.15},
  "storms": ["stm_e5f6g7", "stm_k1l2m3", ...],
  "entities": ["ExchangeX", "LendProtocol", "StableCoinY"],
  "related_harm_records": ["hrm_n4o5p6"]
}
```

`GET` `/fronts/{front_id}/storms` List storms in a front

# 6. Shadow Detector API

The Shadow Detector identifies silent harm — victims who may not even know they are being harmed. These endpoints manage harm records, victim shadow profiles, and the consent unlock mechanism. No victim is ever contacted without consent.

## 6.1 Harm Records

`POST` `/shadow/harm-records` Create a harm record

Request Body:

```
{
  "harm_type": "wage_theft",
  "severity": "high",
  "description": "Platform withheld creator payouts for 90+ days",
  "entity_accused": "CreatorPlatformZ",
  "source_storms": ["stm_e5f6g7"],
  "estimated_affected": 2300,
  "estimated_amount_usd": 1450000.00
}
```

Response:

```
201 Created
{
  "record_id": "hrm_n4o5p6",
  "harm_type": "wage_theft",
  "severity": "high",
  "entity_accused": "CreatorPlatformZ",
  "created_at": "2026-02-06T15:00:00Z"
}
```

`GET` `/shadow/harm-records` List harm records

Path / Query Parameters:

- `harm_type` — Filter by harm type enum

- `entity` — Filter by accused entity
- `severity` — Filter by severity

GET `/shadow/harm-records/{record_id}` Get harm record detail

## 6.2 Victim Shadow Profiles

GET `/shadow/profiles` List shadow profiles

Returns pseudonymized profiles only. No PII is ever returned through the API. Profile identifiers are system-generated pseudonyms.

Path / Query Parameters:

- `status` — detected | monitoring | consent_pending | active | represented
- `harm_type` — Filter by linked harm type

GET `/shadow/profiles/{profile_id}` Get profile detail

Response:

```
200 OK
{
  "profile_id": "vsp_q7r8s9",
  "pseudonym": "shadow-alpha-7",
  "status": "consent_pending",
  "harm_records": ["hrm_n4o5p6"],
  "severity_aggregate": 0.85,
  "estimated_loss_usd": 630.00,
  "consent_id": "cns_t0u1v2",
  "detected_at": "2026-02-06T15:10:00Z"
}
```

## 6.3 Consent Unlock

Consent is the most critical gate in the system. No victim is contacted, represented, or acted upon without explicit consent flowing through this mechanism. Consent can be withdrawn at any time.

`POST` `/shadow/consent` Request consent from a victim

Requires write:hrn scope. The request itself is logged immutably. The system sends a single, non-intrusive notification to the victim via their preferred platform.

Request Body:

```
{
  "profile_id": "vsp_q7r8s9",
  "scope": ["view_evidence", "legal_representation", "claim_filing"],
  "method": "platform_dm",
  "expires_in_days": 30
}
```

Response:

```
201 Created
{
  "consent_id": "cns_t0u1v2",
  "status": "pending",
  "requested_at": "2026-02-06T15:15:00Z",
  "expires_at": "2026-03-08T15:15:00Z"
}
```

`PATCH` `/shadow/consent/{consent_id}` Update consent status

Can be set to: granted, denied, withdrawn. Once withdrawn, all downstream actions are immediately frozen.

Request Body:

```
{
  "status": "granted"
}
```

`GET` `/shadow/consent/{consent_id}` Check consent status

# 7. LITMUS Evaluation API

The LITMUS Financial Reality Test evaluates any entity against six criteria: Lives in bear markets (L), Independent of speculation (I), Tolerates conflict (T), Measures execution (M), Uncomfortable transparency (U), Settles real-world consequences (S). Each criterion is scored by an independent agent.

`POST` `/litmus/evaluations` Trigger a LITMUS evaluation

Evaluations run asynchronously. Six parallel agents score the entity independently. Results are available via GET or webhook.

Request Body:

```
{
  "target_entity": "DeFi Protocol Alpha",
  "context": {
    "website": "https://protocol-alpha.io",
    "chain": "solana",
    "category": "lending"
  }
}
```

Response:

```
202 Accepted
{
  "evaluation_id": "eval_w3x4y5",
  "status": "in_progress",
  "target_entity": "DeFi Protocol Alpha",
  "started_at": "2026-02-06T15:30:00Z",
  "estimated_completion": "2026-02-06T15:35:00Z"
}
```

`GET` `/litmus/evaluations/{eval_id}` Get evaluation result

Response:

```
200 OK
{
  "evaluation_id": "eval_w3x4y5",
  "target_entity": "DeFi Protocol Alpha",
  "composite": 0.72,
  "criteria": [
    {"letter": "L", "label": "Lives in bear markets", "score": 0.85, "confidence": 0.9},
    {"letter": "I", "label": "Independent of speculation", "score": 0.60, ...},
    {"letter": "T", "label": "Tolerates conflict", "score": 0.78, ...},
    {"letter": "M", "label": "Measures execution", "score": 0.70, ...},
    {"letter": "U", "label": "Uncomfortable transparency", "score": 0.65, ...},
    {"letter": "S", "label": "Settles consequences", "score": 0.74, ...}
  ],
  "evaluated_at": "2026-02-06T15:34:22Z"
}
```

GET `/litmus/evaluations` List evaluations

Path / Query Parameters:

- `entity` — Filter by target entity
- `min_score` — Minimum composite score
- `max_score` — Maximum composite score

GET `/litmus/leaderboard` Public LITMUS leaderboard

Public endpoint. No authentication required. Returns entities ranked by composite LITMUS score. Updated every 6 hours.

Path / Query Parameters:

- `category` — Filter by entity category
- `limit` — Number of results (default 25)

# 8. Claims & Cases API

Claims are the core financial accountability unit. Every unpaid dollar, withheld payment, or exploited fee is tracked as a verifiable financial object with a complete audit trail.

`POST` `/claims` File a new claim

Request Body:

```
{
  "respondent_entity": "CreatorPlatformZ",
  "harm_type": "payout_withholding",
  "amount_claimed_usd": 630.00,
  "description": "60-day payout withheld without explanation",
  "related_harm_records": ["hrm_n4o5p6"],
  "evidence": [
    {
      "evidence_type": "financial_record",
      "description": "Payout dashboard screenshot",
      "storage_uri": "gs://evidence/payout_screenshot.png"
    }
  ]
}
```

Response:

```
201 Created
{
  "claim_id": "clm_z6a7b8",
  "status": "filed",
  "respondent_entity": "CreatorPlatformZ",
  "amount_claimed_usd": 630.00,
  "filed_at": "2026-02-06T16:00:00Z"
}
```

`GET` `/claims/{claim_id}` Get claim detail

Path / Query Parameters:

- `claim_id` — UUID of the claim

### GET `/claims` List claims

Path / Query Parameters:

- `status` — filed | under_review | escalated | resolved | closed
- `respondent` — Filter by respondent entity
- `harm_type` — Filter by harm type
- `min_amount` — Minimum claimed amount

### POST `/claims/{claim_id}/escalate` Escalate a claim

Escalation triggers a governance Decision gate. The claim advances only when the three-signature gate (Verifier + Policy + Risk) approves.

Request Body:

```
{
  "reason": "Respondent unresponsive after 14-day notice period",
  "target_status": "escalated"
}
```

### POST `/claims/{claim_id}/evidence` Attach evidence to a claim

Request Body:

```
{
  "evidence_type": "email",
  "description": "Support ticket showing no response",
  "storage_uri": "gs://evidence/support_ticket.pdf"
}
```

### GET `/claims/{claim_id}/execution-log` Get execution history

Returns every action taken on this claim, ordered chronologically. Includes the computed Execution Score.

# 9. Governance & Policy API

All significant actions pass through the governance layer. The 7 Laws are enforced by deterministic policy rules evaluated before any state change. The three-signature decision gate ensures no single agent can act unilaterally.

## 9.1 Policy Rules

`GET` `/governance/rules` List all policy rules

Path / Query Parameters:

- `law` — Filter by law ID (law_1 through law_7)
- `enabled` — true | false

`POST` `/governance/rules` Create a policy rule

Requires admin:governance scope. Rules are version-controlled; previous versions remain in the audit trail.

Request Body:

```
{
  "name": "block-unconsented-contact",
  "law": "law_3_consent_and_privacy",
  "priority": 10,
  "conditions": [
    {"field": "action.type", "operator": "eq", "value": "contact_victim"},
    {"field": "consent.status", "operator": "neq", "value": "granted"}
  ],
  "actions": [
    {"action_type": "block", "target": "shadow_detector"},
    {"action_type": "log", "target": "audit", "params": {"severity": "critical"}}
  ]
}
```

`PATCH` `/governance/rules/{rule_id}` Update a policy rule

Partial updates supported. Changing priority or conditions creates a new version.

## 9.2 Decisions (Three-Signature Gate)

`POST` `/governance/decisions` Create a decision request

Request Body:

```
{
  "context_type": "claim_escalation",
  "context_id": "clm_z6a7b8",
  "requested_by": "agent_case_manager"
}
```

Response:

```
201 Created
{
  "decision_id": "dec_c9d0e1",
  "context_type": "claim_escalation",
  "outcome": "deferred",
  "signatures": []
}
```

`POST` `/governance/decisions/{id}/sign` Submit a signature

Each decision needs three signatures: verifier, policy, risk. Any signer can veto (approved=false, veto=true), which immediately blocks the action regardless of other signatures.

Request Body:

```
{
  "signer": "verifier",
  "approved": true,
  "reason": "Evidence verified, escalation warranted"
}
```

`GET` `/governance/decisions/{id}` Get decision status

`GET` `/governance/audit-log` Query the audit trail

Immutable, append-only log. Every policy evaluation, decision, and state change is recorded here. Eligible for chain anchoring.

Path / Query Parameters:

- `context_type` — Filter by context
- `actor` — Filter by agent or user
- `since` — ISO 8601 lower bound
- `until` — ISO 8601 upper bound

# 10. Human Representation Network API

The HRN bridges algorithmic truth to human justice. Independent legal professionals, auditors, and mediators receive anonymized opportunity briefs and accept cases through this API. All engagement requires prior victim consent.

## 10.1 Opportunity Briefs

`POST` `/hrn/briefs` Publish an opportunity brief

Briefs contain NO personally identifiable information. They are visible to qualified HRN members based on role and jurisdiction.

Request Body:

```
{
  "harm_type": "platform_lockout",
  "severity": "high",
  "jurisdiction_hint": "US-CA",
  "estimated_amount_usd": 45000.00,
  "affected_count": 120,
  "summary": "Platform locked creator accounts without cause...",
  "required_roles": ["legal_advocate"]
}
```

`GET` `/hrn/briefs` List available briefs

Path / Query Parameters:

- `role` — Filter by required role
- `jurisdiction` — Filter by jurisdiction
- `harm_type` — Filter by harm type

## 10.2 Engagements

`POST` `/hrn/engagements` Create an engagement

Requires that the referenced ConsentUnlock is active (granted and not expired). The API validates consent before creating the engagement.

Request Body:

```
{
  "member_id": "hrn_f2g3h4",
  "profile_id": "vsp_q7r8s9",
  "consent_id": "cns_t0u1v2",
  "claim_ids": ["clm_z6a7b8"]
}
```

`PATCH` `/hrn/engagements/{id}` Update engagement status

Request Body:

```
{
  "status": "active",
  "notes": ["Initial consultation completed"]
}
```

`GET` `/hrn/engagements/{id}` Get engagement detail

## 10.3 Members

`GET` `/hrn/members` List HRN members

Path / Query Parameters:

- `role` — Filter by role
- `jurisdiction` — Filter by jurisdiction
- `status` — available | assigned | active

`GET` `/hrn/members/{member_id}` Get member profile

Returns public profile: role, specializations, jurisdiction, cases completed, and success rate. No personal contact info is exposed.

# 11. Event Bus & Webhooks

The internal event bus powers agent-to-agent communication. External consumers can subscribe to events via webhooks. All events carry a correlation ID for tracing across the system.

Event Topics:

| Topic | Description |
|---|---|
| `nie.noise.classified` | Noise event classified |
| `nie.storm.detected` | New storm formed |
| `nie.storm.escalated` | Storm escalated |
| `nie.front.detected` | New front detected |
| `sde.harm.detected` | Harm record created |
| `sde.consent.granted` | Victim consent granted |
| `litmus.eval.completed` | LITMUS evaluation finished |
| `case.claim.filed` | New claim filed |
| `case.claim.escalated` | Claim escalated |
| `gov.decision.resolved` | Decision gate resolved |
| `gov.policy.violation` | Policy rule violated |
| `hrn.engagement.created` | HRN engagement started |

## 11.1 Webhook Subscriptions

`POST` `/webhooks` Create a webhook subscription

Webhook payloads include X-GhostLedger-Signature header for verification using HMAC-SHA256. Deliveries retry 3 times with exponential backoff (10s, 60s, 300s).

Request Body:

```
{
  "url": "https://your-app.com/webhook",
  "topics": ["nie.storm.escalated", "case.claim.filed"],
  "secret": "whsec_...",
  "active": true
}
```

GET /webhooks List webhook subscriptions

DELETE /webhooks/{id} Remove a webhook

Webhook Payload Format:

```
POST https://your-app.com/webhook
Headers:
  X-GhostLedger-Signature: sha256=abc123...
  X-GhostLedger-Event: nie.storm.escalated

Body:
{
  "event_id": "bus_i5j6k7",
  "topic": "nie.storm.escalated",
  "timestamp": "2026-02-06T17:00:00Z",
  "correlation_id": "cor_l8m9n0",
  "payload": { ... }
}
```

# 12. Agent Management API

GhostLedger runs 27 autonomous agents across 6 categories. This API manages agent lifecycle, configuration, and monitoring. All agents operate under the governance layer and are subject to policy rules.

`GET` `/agents` List all agents

Path / Query Parameters:

- `category` — litmus_evaluation | noise_intelligence | shadow_detection | case_management | governance | distribution
- `enabled` — true | false

Response:

```
200 OK
{
  "agents": [
    {
      "agent_id": "agt_o1p2q3",
      "name": "litmus-L-agent",
      "category": "litmus_evaluation",
      "enabled": true,
      "status": "running",
      "last_heartbeat": "2026-02-06T17:29:55Z"
    }
  ]
}
```

`GET` `/agents/{agent_id}` Get agent detail

`PATCH` `/agents/{agent_id}` Update agent config

Requires admin:agents scope. Disabling an agent gracefully drains its current tasks before stopping.

Request Body:

```
{
  "enabled": false,
  "max_concurrent": 3
}
```

### GET `/agents/{agent_id}/metrics` Get agent metrics

Response:

```
200 OK
{
  "agent_id": "agt_o1p2q3",
  "uptime_hours": 720,
  "events_processed": 14523,
  "avg_latency_ms": 245,
  "error_rate": 0.002,
  "last_24h": {
    "processed": 892,
    "errors": 2
  }
}
```

### POST `/agents/{agent_id}/restart` Restart an agent

Triggers a graceful restart. Current tasks are completed before restart.

# 13. Chain Anchor API

The Chain Anchor service records SHA-256 hashes of critical records on the Solana blockchain, creating immutable timestamps that prove a record existed at a specific time. This is the cryptographic backbone of GhostLedger's accountability guarantee.

`POST` `/chain/anchor` Anchor a record on-chain

Requires admin:chain scope. Anchoring is batched every 60 seconds for cost efficiency. Urgent anchors can use priority=true to submit immediately.

Request Body:

```
{
  "record_type": "claim",
  "record_id": "clm_z6a7b8",
  "data_hash": "sha256:a1b2c3d4e5f6..."
}
```

Response:

```
201 Created
{
  "anchor_id": "anc_r3s4t5",
  "tx_signature": "5KtP9...",
  "slot": 234567890,
  "block_time": "2026-02-06T17:45:00Z",
  "program_id": "Ghost1edger...",
  "record_type": "claim",
  "record_id": "clm_z6a7b8"
}
```

`GET` `/chain/anchor/{anchor_id}` Get anchor detail

`GET` `/chain/verify` Verify a record against chain

Public endpoint. Anyone can verify that a record exists on-chain with the correct hash. This is the core transparency mechanism.

Path / Query Parameters:

- `record_type` — Type of the record
- `record_id` — ID of the record
- `data_hash` — Expected SHA-256 hash

Response:

```
200 OK
{
  "verified": true,
  "anchor_id": "anc_r3s4t5",
  "tx_signature": "5KtP9...",
  "anchored_at": "2026-02-06T17:45:00Z",
  "hash_match": true
}
```

### GET `/chain/anchors` List recent anchors

Path / Query Parameters:

- `record_type` — Filter by type
- `since` — ISO 8601 lower bound
- `limit` — Results per page

# 14. Rate Limits & Pagination

Rate Limits:

All endpoints are rate-limited per API key. Limits are returned in response headers:

```
X-RateLimit-Limit: 1000          # requests per window
X-RateLimit-Remaining: 847        # remaining in window
X-RateLimit-Reset: 1707235200     # window reset (Unix epoch)
```

| Scope | Limit | Window |
|---|---|---|
| read:* | 1000 req | 1 minute |
| write:* | 200 req | 1 minute |
| admin:* | 100 req | 1 minute |
| Public endpoints | 60 req | 1 minute |
| Webhook deliveries | 10,000 | per day |

Pagination:

All list endpoints use cursor-based pagination. The cursor is an opaque string; do not parse or construct cursors manually.

```
GET /v1/noise/events?limit=50&cursor=cur_abc123

Response:
{
  "events": [ ... ],
  "cursor": "cur_def456",       // null if no more results
  "has_more": true,
  "total": 1247
}
```

Idempotency:

All POST endpoints support the X-Idempotency-Key header. If a request is retried with the same idempotency key within 24 hours, the server returns the original response without creating a duplicate

resource.

```
POST /v1/claims
X-Idempotency-Key: idem_unique_key_123

// Retry with same key returns 200 with original claim
// instead of creating a duplicate
```

---

This specification covers all public-facing endpoints of the GhostLedger system. Internal agent-to-agent communication flows through the event bus and is not exposed externally. For the complete data schema definitions, see the Engineering Specifications document and ghostledger_types.py.

*"The system does not speak unless it has something to verify."*