BH Audit Schema: An Open Standard for PHI-Safe Audit Logging in Behavioral Health Systems

Tanmaya Kumar
Behavioral Health Open Source (bh-healthcare.org)
https://github.com/bh-healthcare/bh-audit-schema

April 2026

Abstract

Behavioral health systems face audit logging challenges that general-purpose healthcare standards do not adequately address. Regulations such as 42 CFR Part 2 impose stricter confidentiality requirements for substance use disorder treatment records than standard HIPAA provisions, and mental health data carries elevated stigma risk if exposed through audit trails. Existing audit approaches tend to capture excessive protected health information (PHI) in log entries or lack the structure needed for meaningful compliance analysis. This paper presents BH Audit Schema, an open-source, versioned JSON Schema standard for behavioral health audit events. The contribution is the specific combination: a domain-specific audit contract that is machine-validatable, PHI-minimizing by construction, and explicitly mapped to HIPAA Security Rule, 42 CFR Part 2, and SOC 2 control objectives. The schema records the actions that occurred and the resource types they acted on, and it does so without logging any underlying clinical content. This paper describes the schema's design principles, threat model, specification, regulatory control mappings, and reference implementation deployed in a production behavioral health platform. The standard, tooling, and documentation are publicly available under the Apache 2.0 license at https://github.com/bh-healthcare/bh-audit-schema.

Keywords: audit logging, behavioral health, HIPAA, PHI, 42 CFR Part 2, JSON Schema, compliance, open source

1 Introduction

Behavioral health data is among the most sensitive categories of protected health information. Therapy notes contain deeply personal disclosures. Substance use disorder records carry legal protections beyond standard HIPAA. Mental health diagnoses are associated with significant social stigma, and crisis intervention records have direct life-safety implications. A breach or unauthorized access to this data can cause profound harm to patients, including damage to relationships, employment discrimination, and personal distress that may exacerbate the very conditions being treated.

Behavioral health organizations face a fragmented landscape when implementing audit logging. Large healthcare systems can afford custom audit infrastructure, but many small and mid-sized behavioral health providers, who deliver a substantial share of community-based care, lack dedicated security engineering resources. Legacy systems have inconsistent or missing audit capabilities. Custom implementations create non-portable audit formats, and vendor diversity means no common audit event structure exists across tools.

The result is a significant implementation gap. Organizations either fail to meet audit requirements or invest disproportionate resources reinventing audit logging for each system. The

regulatory landscape continues to grow more complex at the same time, with HIPAA Security Rule requirements intersecting with the stricter 42 CFR Part 2 provisions governing substance use disorder records, state-specific mental health confidentiality laws, and contractual payer requirements.

This paper presents BH Audit Schema, an open-source standard designed to address this gap. The contribution is the combination of four properties in one artifact: a behavioral-health-specific audit contract, machine-validatable through JSON Schema, PHI-minimizing by construction, and explicitly mapped to HIPAA Security Rule, 42 CFR Part 2, and SOC 2 control objectives. Each of these properties exists individually in other standards and tools, but their combination is what behavioral health organizations need and what has not previously been available as an open-source artifact.

This paper is written for engineers and security practitioners building or evaluating healthcare infrastructure, particularly applied health informatics and systems teams working in behavioral health. Compliance, regulatory, and clinical informatics readers will find the control mappings and regulatory sections useful, but the primary lens is engineering: how to design audit logging that is simultaneously compliant, privacy-safe, and practical to adopt. The schema provides a versioned, machine-validatable contract for audit events that enforces PHI safety by default. It defines *what* to log and leaves *how* to implement logging as an adopter decision. The schema is accompanied by a Python reference implementation (bh-audit-logger), FastAPI middleware (bh-fastapi-audit), and documentation mapping schema fields to specific regulatory control objectives.

The remainder of this paper is organized as follows. Section 2 surveys the regulatory background and related work. Section 3 presents the schema's design principles. Section 4 describes the threat model the schema defends against. Section 5 describes the schema specification. Section 6 maps the schema to regulatory control objectives. Section 7 discusses the reference implementation and limited production observations. Section 8 discusses limitations and future work, and Section 9 concludes.

2 Background and Related Work

2.1 Regulatory Requirements for Behavioral Health Audit Logging

Organizations handling behavioral health data operate under multiple overlapping regulatory frameworks, each imposing audit trail requirements.

The **HIPAA Security Rule** (§164.312(b)) requires covered entities to "implement hardware, software, and/or procedural mechanisms that record and examine activity in information systems that contain or use electronic protected health information (ePHI)." Additional provisions address access controls (§164.312(a)(1)), unique user identification (§164.312(a)(2)(i)), information system activity review (§164.308(a)(1)(ii)(D)), login monitoring (§164.308(a)(5)(ii)(C)), and integrity controls (§164.312(c)(1)).

42 CFR Part 2 governs the confidentiality of substance use disorder (SUD) patient records and imposes requirements substantially stricter than general HIPAA. Section 2.13(a) establishes the general rule that patient records may be used or disclosed only as permitted by the regulations, and Section 2.31 specifies the written consent form requirements that authorize most disclosures. Section 2.16 mandates formal policies and procedures to protect SUD record security. These provisions create a regulatory environment where audit trails for behavioral health systems must be more granular and more privacy-conscious than those required for general healthcare.

SOC 2 Trust Services Criteria provide a framework for evaluating controls at service organizations, with criteria covering logical access (CC6.1), role-based access (CC6.3), system monitoring (CC7.2), and security event evaluation (CC7.3). Many behavioral health technology

vendors undergo SOC 2 audits as a condition of payer contracts.

2.2 Existing Approaches

Several standards address audit logging in healthcare, though none specifically targets the behavioral health domain.

The **IHE Audit Trail and Node Authentication (ATNA)** integration profile defines audit message formats based on RFC 3881 and DICOM, primarily targeting hospital information systems and imaging workflows. ATNA is comprehensive for its intended domain but does not address behavioral health-specific concerns such as 42 CFR Part 2 consent gating or the elevated PHI sensitivity of mental health records.

FHIR AuditEvent (R4/R5) provides a resource type for recording audit information within the FHIR ecosystem. It offers a flexible, extensible structure but requires adopters to define their own constraints, field semantics, and PHI-safety policies. For small behavioral health organizations without FHIR expertise, this flexibility can become a barrier.

Cloud provider audit services (AWS CloudTrail, Azure Activity Log, GCP Cloud Audit Logs) capture infrastructure-level events but lack clinical context. They record that an API call was made to a database but cannot distinguish a routine appointment lookup from access to a patient's substance use disorder treatment record.

General-purpose logging frameworks (ELK Stack, Splunk, structured logging libraries) provide transport and storage but impose no schema constraints. Organizations using these tools must still define what constitutes an audit event, what fields are required, and how to prevent PHI from entering log entries.

The BH Audit Schema sits between these approaches. It is domain-specific enough to encode behavioral health regulatory requirements, schema-validated enough to enforce consistency, and lightweight enough for small organizations to adopt without specialized infrastructure. Table 1 summarizes how BH Audit Schema compares to existing approaches across the properties most relevant to behavioral health deployments.

Table 1: Comparison of audit logging approaches across properties relevant to behavioral health deployments. ✓ indicates the property is native to the approach. ∼ indicates partial support that depends on adopter configuration. ✗ indicates the property is absent or explicitly out of scope.

Property	ATNA	FHIR AuditEvent	CloudTrail / infra	Generic logs	BH Audit
Behavioral-health-specific	✗	✗	✗	✗	✓
PHI-minimizing by default	✗	∼	✗	✗	✓
Schema validation	✓	✓	✗	✗	✓
Clinical action semantics	∼	✓	✗	✗	✓
Consent-gating support	✗	∼	✗	✗	✓
Adoption path for non-FHIR orgs	∼	✗	✓	✓	✓

3 Design Principles

The schema is governed by five design principles, each motivated by the operational realities of behavioral health technology.

3.1 PHI Safety by Default

The most important design decision is that audit events capture *what happened* and *to what*, without recording *the content* of what was accessed. This principle manifests in several concrete rules:

- **No bodies by default.** Events record resource types and identifiers (e.g., "Note `note_456` was accessed") but never clinical text, diagnosis codes, or treatment details.
- **Identifiers over identities.** The schema uses opaque identifiers (`user_123`, `pat_456`) rather than human-readable names, email addresses, or other directly identifying attributes.
- **Templates over raw paths.** HTTP route information uses parameterized templates (`/patients/{patient_id}/notes/{note_id}`) rather than actual request paths that would embed identifiers.
- **Sanitized error messages.** Error messages must not contain PHI, even in failure scenarios. "Access denied" is acceptable. "User cannot access patient John Smith's note" would violate this rule.
- **Safe metadata.** The extensible `metadata` object is constrained to scalar values with a maximum of 20 properties, preventing nested structures that could harbor PHI.

This design ensures that audit logs do not become a secondary PHI repository. Log aggregation and analysis tools do not require PHI access, and breach scope is reduced if audit systems are compromised.

3.2 Strict Validation

The schema uses JSON Schema with `additionalProperties: false` at all levels (except the `metadata` object) to ensure events conform exactly to the specification. This prevents the gradual accumulation of unvalidated fields that, in practice, often become vectors for PHI leakage. Combined with required field constraints and enumerated value sets, strict validation makes schema compliance a binary, testable property.

3.3 Implementation Flexibility

The schema is strict about *what* to log and deliberately silent about *how*. Organizations can use any programming language, framework, or storage backend. They can implement synchronous or asynchronous logging, choose between relational databases, document stores, or log aggregation services, and integrate with existing SIEM infrastructure. This flexibility is essential for adoption in a domain where technology stacks vary widely.

3.4 Explicit Classification

The schema includes explicit classification fields so consumers do not have to infer whether an event involved sensitive data. The boolean `action.phi_touched` flag indicates whether an action accessed or modified PHI, and `action.data_classification` provides more granular categorization (`PHI`, `PII`, `NONE`, `UNKNOWN`). These fields are optional in the schema (`data_classification`

defaults to `UNKNOWN`) so that baseline adoption is not blocked, but compliance mappings assume adopters populate them for events touching regulated data. These fields enable risk-based handling of audit events without requiring consumers to maintain mappings of resource types to sensitivity levels.

3.5 Versioned Evolution

Every audit event includes a `schema_version` field, and the schema follows semantic versioning. Version 1.x releases are backward-compatible additions. Breaking changes require a major version increment. Immutable version snapshots are maintained in the repository alongside the latest schema, supporting environments where audit records must be retained and validated over multi-year horizons.

4 Threat Model

The schema's design principles are grounded in a specific set of risks that audit logging can introduce into a healthcare system. Before describing the schema itself, this section makes those risks explicit so the design decisions can be evaluated against what they are intended to prevent.

Logs as a secondary PHI repository. The most significant risk is that audit logs become a parallel store of clinical data. An engineer reviewing a log entry to debug a failed request should not be able to read clinical notes, diagnosis codes, or treatment details from that log entry. If logs contain PHI, every log aggregation system, every search index, every developer workstation that accesses logs becomes part of the compliance boundary. This inflates breach scope and complicates HIPAA and 42 CFR Part 2 obligations.

Leakage through request and response bodies. A common antipattern is logging raw HTTP request bodies or response payloads to help with debugging. In a behavioral health context, this captures exactly the content that should not be in logs. The schema prohibits this category of data by defining no fields that accept clinical content and constraining extensible metadata to scalar values.

Leakage through raw URL paths and query strings. Paths like `/patients/12345/notes/67890` embed patient and record identifiers that may be sensitive. More concerning are query strings that encode search terms, filter parameters, or free-text inputs. The schema's use of parameterized route templates (`/patients/{patient_id}/notes/{note_id}`) rather than raw request paths prevents this leakage pattern by construction.

Leakage through error messages. Exception messages and error strings often concatenate PHI into human-readable text ("User cannot access patient Jane Doe's note dated 2024-11-03"). The schema enforces that error messages be sanitized, and the reference implementation provides helpers that generate compliant error descriptions.

Leakage through nested or free-form metadata. Extensible metadata fields are necessary for domain-specific context but are also the most common place where PHI re-enters audit records. The schema constrains metadata to scalar key-value pairs with a bounded number of properties, preventing nested structures or large text blobs that could carry clinical content.

Inconsistent audit semantics across services. When each service defines its own audit format, correlation and analysis become brittle. A security event affecting multiple services produces audit records with different field names, different value conventions, and different levels of detail. Investigators cannot construct a coherent timeline. The schema addresses this by providing a single contract that all services emit against, which turns cross-service audit analysis from a parsing problem into a query problem.

Threats outside scope. The schema does not defend against compromise of the audit system itself, insider threats with direct database access, or malicious modification of audit records once they are stored. Integrity controls (event hashing, hash chaining) provide tamper-

evidence but not tamper-prevention. Organizations must still protect the audit storage layer through access controls, retention policies, and monitoring appropriate to the sensitivity of their environment.

5 Schema Specification

The BH Audit Schema v1.1 defines a JSON object with seven required top-level sections and three optional sections. This section describes each component.

5.1 Event Envelope

Every event includes three envelope fields:

- `schema_version` (string, required): The schema version this event conforms to. Enables consumers to select the appropriate validation schema.
- `event_id` (string, UUID format, required): A globally unique identifier for the event. The v1.1 schema enforces UUID format to ensure uniqueness across distributed systems.
- `timestamp` (string, date-time format, required): The time the event occurred, in ISO 8601 format.

5.2 Service

The `service` object identifies the system that generated the event:

- `service.name` (string, required): The name of the producing service (e.g., `bh-intake-api`).
- `service.environment` (string, optional): The deployment environment (e.g., `prod`, `staging`).

5.3 Actor

The `actor` object identifies who performed the action:

- `actor.subject_id` (string, minLength: 1, required): The unique identifier for the actor.
- `actor.subject_type` (enum: `human`, `service`, required): Distinguishes human users from automated service principals.
- `actor.roles` (array of strings, optional): The roles held by the actor at the time of the action. Bounded with non-empty items in v1.1.
- `actor.org_id` (string, optional): The organization the actor belongs to.
- `actor.owner_org_id` (string, optional, v1.1): The organization that owns the resource being accessed. Enables detection of cross-organization access in multi-tenant environments.

5.4 Action

The `action` object describes what was done:

- `action.type` (enum, required): The category of action. Enumerated values include `READ`, `CREATE`, `UPDATE`, `DELETE`, `EXPORT`, `LOGIN`, `LOGOUT`, `PRINT`, and `OTHER`.
- `action.name` (string, optional): A more specific action identifier for domain-specific actions.

- `action.phi_touched` (boolean, optional): Whether the action accessed or modified protected health information.
- `action.data_classification` (enum, optional, default `UNKNOWN`): The sensitivity classification of the data involved (`PHI`, `PII`, `NONE`, `UNKNOWN`).

5.5 Resource

The `resource` object identifies the target of the action:

- `resource.type` (string, required): The category of resource (e.g., `Patient`, `Note`, `Encounter`).
- `resource.id` (string, optional): The resource's unique identifier.
- `resource.patient_id` (string, optional): The patient associated with the resource, enabling patient-centric access review.

5.6 Outcome

The `outcome` object records the result:

- `outcome.status` (enum, required): `SUCCESS`, `FAILURE`, or `DENIED`. The v1.1 addition of `DENIED` distinguishes authorization-based access denials from operational failures, a critical distinction for compliance analysis.
- `outcome.error_type` (string, conditionally required): Required when status is `FAILURE` or `DENIED`. Categorizes the failure or denial reason (e.g., `RoleDenied`, `CrossOrgAccessDenied`, `ConsentRequired`).
- `outcome.error_message` (string, optional): A sanitized, PHI-free error description.

5.7 Optional Sections

Three optional sections provide additional context:

- `correlation`: Contains `request_id`, `trace_id`, and `session_id` for distributed tracing across microservice architectures.
- `http`: Captures HTTP-specific context including `method`, `route_template`, `status_code`, `client_ip`, and `user_agent`. The `client_ip` field supports both IPv4 and IPv6 formats.
- `integrity`: Supports tamper detection with `event_hash`, `prev_event_hash` (for chain verification), and `hash_alg` (constrained to `sha256`, `sha384`, or `sha512`).

5.8 Example Event

Listing 1 shows a minimal compliant event.

Listing 1: A minimal BH Audit Schema v1.1 event recording a care coordinator reading a patient note.

```
{
  "schema_version": "1.1",
  "event_id": "6d3f0f6b-0c1a-4b9f-9d6f-9f6f7f5b2b0a",
  "timestamp": "2026-01-06T18:40:12Z",
  "service": {
    "name": "bh-intake-api",
    "environment": "prod"
```

```
  },
  "actor": {
    "subject_id": "user_123",
    "subject_type": "human",
    "roles": ["care_coordinator"]
  },
  "action": {
    "type": "READ",
    "phi_touched": true,
    "data_classification": "PHI"
  },
  "resource": {
    "type": "Note",
    "id": "note_456",
    "patient_id": "pat_789"
  },
  "outcome": {
    "status": "SUCCESS"
  }
}
```

Note that the event captures the fact that a care coordinator read a clinical note belonging to a specific patient, without revealing any clinical content, the patient's name, or the care coordinator's identity beyond an opaque identifier.

6 Controls Mapping

One of the key contributions of this work is the explicit mapping between schema fields and regulatory control objectives. This section summarizes the mappings. The full mapping document is maintained in the project repository.

6.1 HIPAA Security Rule

Table 2 maps schema fields to the HIPAA Security Rule provisions most relevant to audit logging.

Table 2: Mapping of BH Audit Schema fields to HIPAA Security Rule provisions.

HIPAA Provision	**Supporting Schema Fields**
§164.312(b) Audit Controls	`event_id`, `timestamp`, `actor.subject_id`, `action.type`, `resource.*`, `outcome.status`, `action.phi_touched`
§164.312(a)(1) Access Control	`outcome.status: DENIED`, `outcome.error_type`, `actor.roles`, `actor.org_id`, `actor.owner_org_id`
§164.312(a)(2)(i) Unique User ID	`actor.subject_id` (minLength: 1), `actor.subject_type`
§164.308(a)(1)(ii)(D) Activity Review	`correlation.*`, `http.*`, `metadata`
§164.308(a)(5)(ii)(C) Login Monitoring	`action.type: LOGIN`, `outcome.status`, `outcome.error_type`, `http.client_ip`
§164.312(c)(1) Integrity	`integrity.event_hash`, `integrity.prev_event_hash`, `integrity.hash_alg`

6.2 42 CFR Part 2

The schema provides specific support for 42 CFR Part 2 requirements through several mechanisms. The `action.data_classification: PHI` flag identifies events touching regulated substance use disorder data. The `outcome.error_type: ConsentRequired` value indicates access denials due to missing patient consent, directly supporting the Part 2 consent requirements

under §2.13 and §2.31. The `actor.owner_org_id` field enables detection of cross-organization SUD record access attempts, and the scalar-only constraint on `metadata` prevents raw clinical content from entering audit logs.

The schema provides the audit trail infrastructure for consent-gated access but does not model consent documents themselves. Consent tracking (consent identifiers, consent purpose, break-the-glass scenarios) is planned for a future schema version.

6.3 SOC 2 Trust Services Criteria

The schema maps to SOC 2 criteria including CC6.1 (logical access controls via the actor, action, resource, and outcome sections), CC6.3 (role-based access via `actor.roles` and role-denial tracking), CC7.2 (system monitoring via failure and denial pattern detection with correlation fields), and CC7.3 (security event evaluation via action categorization, PHI impact assessment, and incident correlation).

7 Reference Implementation and Deployment

7.1 bh-audit-logger

The primary reference implementation is `bh-audit-logger`, a Python package available on PyPI. The package provides a cloud-agnostic audit event emitter with the following characteristics:

- Structured event construction with compile-time type checking via Python dataclasses.
- Built-in JSON Schema validation against the v1.1 schema.
- Pluggable sink architecture supporting stdout, file, and DynamoDB backends.
- Optional CLI interface for event generation and validation.
- Zero required dependencies for the core library; optional extras for DynamoDB (`boto3`), JSON Schema validation (`jsonschema`), type checking (`mypy`), and development tooling.

7.2 bh-fastapi-audit

A companion package, `bh-fastapi-audit`, provides FastAPI middleware that automatically emits audit events for HTTP requests. It demonstrates how to integrate audit logging into a web application framework, populating schema fields from HTTP context (method, route template, status code, client IP) and delegating PHI classification to application-layer configuration.

7.3 Limited Production Observations

This paper does not present BH Audit Schema as a telemetry study or adoption benchmark. The project is designed for privacy-sensitive behavioral health environments, where audit tooling must avoid becoming a secondary store of protected information. Optional telemetry is disabled by default in open-source usage and, when enabled, is intentionally limited to narrow operational signals such as event counts, validation outcomes, and runtime context metadata. The logger does not inspect underlying business payloads, request bodies, response bodies, or protected clinical detail beyond the schema-safe event envelope. This is a design feature, not a limitation. The goal is to establish a machine-validatable, PHI-minimizing audit contract, not to maximize observability depth.

For this reason, the deployment figures reported below should be interpreted as limited production observations from internal use of the reference implementation. They demonstrate that the schema and middleware can be integrated into real services and can emit valid audit

events reliably under live operating conditions. They do not constitute ecosystem-wide adoption data, a formal evaluation dataset, or evidence of comparative security performance.

The schema and reference implementation are deployed in production at Overstory Health, a McLean Hospital-validated behavioral health platform serving over 100 patients across intake, clinical documentation, care coordination, patient communication, and treatment planning workflows. The deployment uses an AWS serverless architecture with Lambda functions emitting audit events to DynamoDB via the `bh-audit-logger` package. Events are validated against the v1.1 schema at emission time, which ensures that non-compliant events are rejected before storage.

In the first four days of initial production rollout across two services, a patient intake API and a notification service spanning SMS, email, and scheduled communication handlers, the deployment emitted 75,501 audit events across 247 unique Lambda execution contexts. No schema validation failures were observed during this window. The intake service emitted 35 events across 4 deployment instances. The notification service, which handles all outbound patient communications, emitted 75,466 events across 243 deployment instances, reflecting the higher operational volume of the communication pathway relative to the intake pathway.

An opt-in telemetry receiver aggregates emission metadata (event counts, deployment identifiers, service names) without collecting the audit event payloads themselves. This design preserves the PHI-safety properties of the schema while enabling deployment health monitoring. The telemetry pipeline also exposed one real-world operational signal during the initial rollout: a CloudWatch invocation alarm triggered when the notification service ramp exceeded 1,000 invocations within a 24-hour window, illustrating that realistic production volumes can exceed conservative baseline thresholds set during development.

8 Limitations and Future Work

Several limitations of the current work merit discussion.

Single-organization validation. The schema has been deployed in production within a single organization, though it has been validated across multiple services with distinct operational characteristics and event volumes. Multi-organization adoption would provide stronger evidence of the schema's generalizability across different organizational structures, technology stacks, and regulatory jurisdictions. Because telemetry is privacy-constrained, opt-in, and disabled by default for open-source usage, the project does not currently provide broad adoption or performance data across external deployments.

Consent modeling. The current schema supports consent-gated access denial tracking (`ConsentRequired` error type) but does not model consent documents themselves. A future version will introduce optional consent fields (consent identifier, consent purpose, break-the-glass justification) to support the full 42 CFR Part 2 consent lifecycle.

Analytics and alerting patterns. The schema enables structured querying, but this paper does not present specific analytics queries or alerting rules for detecting anomalous access patterns. Reference query patterns are documented in the project repository, and a formal analysis of detection capabilities is planned for future work.

Performance characterization. We do not present quantitative performance benchmarks for event emission, validation, or storage. In practice, the overhead of JSON Schema validation is negligible relative to the latency of the clinical operations being audited, but formal benchmarking would strengthen the implementation guidance.

Interoperability with FHIR AuditEvent. A mapping between BH Audit Schema events and FHIR AuditEvent resources would enable organizations using FHIR-based systems to adopt the schema alongside their existing audit infrastructure. This mapping is planned but not yet implemented.

9 Conclusion

Behavioral health systems operate under regulatory requirements that demand structured, comprehensive audit logging and require heightened privacy protections for the data being audited at the same time. The BH Audit Schema addresses this tension through a design that enforces PHI safety by default, provides explicit mappings to HIPAA, 42 CFR Part 2, and SOC 2 control objectives, and maintains strict validation to prevent schema drift.

Publishing the schema, reference implementations, and compliance documentation as open-source artifacts is intended to reduce the barrier to compliant audit logging for behavioral health organizations of all sizes. The schema is not a compliance solution in itself. It provides the engineering primitives that compliance programs require.

The schema, documentation, reference implementations, and examples are available at https://github.com/bh-healthcare/bh-audit-schema under the Apache 2.0 license. The Python reference implementation is available on PyPI as `bh-audit-logger`.

Acknowledgments

This work was developed independently by the author under the Behavioral Health Open Source organization. The schema has benefited from practical deployment experience at Overstory Health.

Disclaimer

This paper and the associated schema provide engineering standards for audit event structure. They do not constitute legal advice, compliance certification, or guarantee of regulatory compliance. Organizations must conduct their own compliance assessments with qualified professionals.

References

[1] U.S. Department of Health and Human Services. *HIPAA Security Rule*, 45 CFR Part 164, Subpart C. https://www.hhs.gov/hipaa/for-professionals/security/

[2] Substance Abuse and Mental Health Services Administration. *42 CFR Part 2: Confidentiality of Substance Use Disorder Patient Records*. https://www.ecfr.gov/current/title-42/chapter-I/subchapter-A/part-2

[3] American Institute of CPAs. *SOC 2 Trust Services Criteria*. https://us.aicpa.org/interestareas/frc/assuranceadvisoryservices/trustservicescriteria

[4] IHE IT Infrastructure Technical Framework. *Audit Trail and Node Authentication (ATNA)*. https://profiles.ihe.net/ITI/TF/Volume1/ch-9.html

[5] HL7 FHIR. *AuditEvent Resource (R4)*. https://www.hl7.org/fhir/auditevent.html

[6] Wright, A., Andrews, H., Hutton, B. *JSON Schema: A Media Type for Describing JSON Documents*. Internet Engineering Task Force, 2022. https://json-schema.org/specification

[7] Substance Abuse and Mental Health Services Administration. *Confidentiality of Patient Records for Alcohol and Other Drug Treatment*. Technical Assistance Publication (TAP) Series 13. SAMHSA, 2020.

[8] Corrigan, P.W., Druss, B.G., Perlick, D.A. "The Impact of Mental Illness Stigma on Seeking and Participating in Mental Health Care." *Psychological Science in the Public Interest*, 15(2):37–70, 2014.

[9] Kruse, C.S., Kristof, C., Jones, B., Mitchell, E., Martinez, A. "Barriers to Electronic Health Record Adoption: A Systematic Literature Review." *Journal of Medical Systems*, 40:252, 2016.

[10] Rindfleisch, T.C. "Privacy, Information Technology, and Health Care." *Communications of the ACM*, 40(8):92–100, 1997.