FERPA AUDIT EVIDENCE PACK · v1.0 · 2026-04-21

Auditor-ready FERPA evidence pack. One source of truth.

Every PII field, every access point, every rate limit, every audit-log row, every encryption guarantee — mapped to the file path that backs it. Your IT director and counsel can grep our source repo and verify each claim line-by-line.

Pinnova FERPA Audit — Evidence Pack

Document version: v1.0 — 2026-04-21 **Owner:** Jenavus LLC, Privacy Engineering (jpeterson@jenavus.com) **Audience:** District IT directors, compliance officers, counsel **Companion docs:** /privacy (public-facing policy) · /ferpa (FERPA landing page)

> **Purpose:** Single source of truth that catalogs every FERPA-protected field stored by Pinnova, every API access point that can read it, the rate limits + audit-log row + encryption guarantee that govern the access. If any line below drifts from the implementation, this doc must be updated. Auditors should be able to grep the Pinnova source repo for the file paths cited and verify each claim line-by-line.

---

1. Scope

This audit covers the **multi-tenant production** code path (`MULTI_TENANT_V2=true`). The legacy single-tenant JSON-backed code path is in scope only for the `parents.json` and `students.json` fixtures — which contain SYNTHETIC data only and are never used for paying districts.

Statutes / frameworks referenced: - **FERPA** — 20 U.S.C. § 1232g, 34 C.F.R. Part 99 - **§99.31(a)(1)(i)(B)** "school official with legitimate educational interest" - **§99.32** record-of-access requirement - **State analogs** — VA Code § 22.1-287.04, MD § 7-105, NC GS 115C-402.5

2. PII Field Catalog

Every field below stores or derives FERPA-protected student/parent education-record data. Table-by-table inventory:

2.1 `students` table (backend/models/student.py)

COLUMN	TYPE	SENSITIVITY	NOTES
`id`	UUID PK	Internal — not PII	Random; never returned to parents
`district_id`	UUID FK	Tenant key	All reads MUST filter by this
`legacy_id`	varchar(64)	Pseudo-ID	Treated as PII by default
`name`	varchar(255)	**PII**	Full name
`grade`	int	**PII**	Education record
`grade_band`	varchar(16)	**PII**	ES/MS/HS
`student_type`	varchar(16)	**PII**	GEN/SN
`dob`	date	**PII (high)**	Auth factor — never returned to parents
`route_legacy_id`	varchar(64)	**PII**	Education record (transportation)
`stop_id`	varchar(64)	**PII**	Pickup point
`stop_type`	varchar(32)	**PII**	curbside/etc
`pickup_time`	varchar(16)	**PII**	Schedule
`home_address`	varchar(512)	**PII (high)**	Directory information
`lat` / `lng`	float	**PII**	Geo of home address
`hub`	varchar(64)	**PII**	Region designator

COLUMN	TYPE	SENSITIVITY	NOTES
`emergency_contact`	JSON	**PII (high)**	Parent name + phone + relationship
`authorized_pickups`	JSON	**PII**	Names of authorized adults
`raw`	JSON	**PII (high)**	Full source-row passthrough

2.2 `parents` table (backend/models/parent.py)

COLUMN	TYPE	SENSITIVITY	NOTES
`id`	UUID PK	Internal	
`district_id`	UUID FK	Tenant key	
`legacy_id`	varchar(64)	Pseudo-ID	
`name`	varchar(255)	**PII**	Parent/guardian name
`email`	varchar(320)	**PII**	Contact channel
`phone`	varchar(32)	**PII**	Contact channel
`relationship`	varchar(64)	**PII**	Relationship to student

2.3 `users` table (backend/models/user.py) — NOT FERPA

District staff accounts. Personal info but not student-record PII — covered by SOC2 / general privacy, not FERPA.

2.4 `dispatches`, `incidents`, `buses`, `routes` tables

Operational data. `incidents.subject_student` can carry a student legacy_id when an incident references a specific student → treated as PII, audit-logged on read.

3. Access Points (the only routes that can read PII)

Inventory method: grep for any tenant_query() over Student / Parent / Incident.subject_student. Every match below; if the implementation grows, add the row before merging.

3.1 `POST /api/tenant/parent/lookup` (backend/routes/tenant_api.py:225)

ASPECT	IMPLEMENTATION
Auth factor	student_legacy_id + dob (exact match required)
Tenant scope	`tenant_query(Student).filter(legacy_id == student_id)` — physically cannot return cross-district rows
Rate limit	**5 / minute per IP** (rate_limit.py `IP_LIMIT`) **AND 20 / hour per `(student_id, district_id)`** `SUBJECT_LIMIT` — both buckets consulted; whichever trips first returns 429
Failed-attempt handling	Counts toward both buckets; 401 returned (generic — no enumeration)
Audit row	`FerpaAuditLog` `subject_type='student'` `action='read'` (success) or `action='lookup_failed'` (missing student / wrong DOB)
Rate-limit-trip side effect	Additional `AuditLog(event='parent_lookup.rate_limited')` row with `details={reason, retry_after}` so admins surface brute-force in real time
Returned fields	`student_id` `name` `grade` `stop_id` `pickup_time` `route_legacy_id` — **dob is NEVER returned**, **home_address is NEVER returned**, **emergency_contact is NEVER returned**

3.2 `GET /api/tenant/driver/<driver_id>/manifest` (backend/routes/tenant_api.py:194)

ASPECT	IMPLEMENTATION
Auth	JWT bearer (driver role) OR X-Admin-Secret
Tenant scope	`tenant_query(Dispatch + Student)` — driver can only see students on routes they're currently dispatched on

ASPECT	IMPLEMENTATION
Rate limit	None directly (auth-gated); upstream nginx limits apply
Audit row	`FerpaAuditLog` one row per student in the returned list, `subject_type='student'` `action='read'`
Returned fields	`student_id` `name` `grade` `stop_id` `pickup_time` — `dob` `home_address` `emergency_contact` NOT returned

3.3 `GET /api/tenant/dispatcher/incidents` (backend/routes/tenant_api.py:158)

ASPECT	IMPLEMENTATION
Auth	JWT bearer (dispatcher / admin role) OR X-Admin-Secret
Tenant scope	`tenant_query(Incident)` — district-filtered
Rate limit	Auth-gated only
Audit row	`subject_student` field read writes a FerpaAuditLog if non-null (NOT YET WIRED — see §6 followups)
Returned fields	`incident_type` `severity` `subject_bus` `subject_route` `summary` `resolved` — `subject_student` returned ONLY to dispatchers/admins, never to drivers/parents

3.4 Superadmin reads (`/api/admin/districts/<slug>/ferpa-audit`)

ASPECT	IMPLEMENTATION
Auth	`X-Admin-Secret` header (Jenavus-only)
Tenant scope	Cross-district by design — for support/incident response
Rate limit	None (gated by knowing the secret)

ASPECT	IMPLEMENTATION
Audit row	Reads of the audit log do NOT themselves write to the audit log (would be infinite-loop pattern); each superadmin call writes an `AuditLog(event='superadmin.audit_read')` row to the general audit table
Returned fields	Audit metadata only — never the underlying PII the audit refers to

3.5 District admin reads (`/api/tenant/admin/ferpa-audit`)

ASPECT	IMPLEMENTATION
Auth	`require_district_admin` (JWT role=admin OR trusted X-Admin-Secret)
Tenant scope	Single district via `tenant_query(FerpaAuditLog)`
Returned fields	Audit metadata for THIS district only

4. Encryption-at-rest Guarantees

LAYER	TECH	KEY MANAGEMENT
Disk	DigitalOcean Managed Postgres TDE (AES-256, automatic)	DO-managed KMS, rotated by DO per their schedule
Backup	Encrypted snapshots, 7-day retention	Same DO-managed KMS
Transit (client ↔ DB)	TLS 1.2+, certificate-validated	DO-issued
Transit (parent ↔ Pinnova)	TLS 1.2+ enforced by Cloudflare front + DO LB	Cloudflare-managed cert, auto-rotated
App-layer	None additional — TDE is the canonical control	n/a

Verification path for auditors: - DO Managed Database product page documents TDE-on-default for the PG plan. - Cluster ID `cbf1a021-a2c8-4a0b-8d2b-79da2fd7cf2f` is the prod cluster. - Connection string template in `backend/db.py` enforces sslmode (DO-injected).

5. Audit-Log Row Inventory

Two tables hold the access trail:

5.1 `ferpa_audit_logs` (backend/models/audit_log.py:25)

Dedicated FERPA trail. **Required by FERPA §99.32.**

```
id              UUID PK
district_id     UUID FK (CASCADE on district delete EXCEPT see retention §7)
user_id         UUID FK (SET NULL on user delete)
subject_type    'student' | 'parent'
subject_id      legacy_id of the subject (varchar(64))
action          'read' | 'lookup_failed' | 'list' | 'export'
route           HTTP path (helps identify which endpoint)
ip              INET (client IP)
details         JSON (request_id, etc.)
ts              timestamp
```

Indexed on `district_id`, `user_id`, `subject_id`, `ts`.

Write helper: `_ferpa_audit(subject_type, subject_id, action)` in `backend/routes/tenant_api.py:46`. Best-effort (never blocks the request); a write failure is logged with `logger.warning`.

5.2 `audit_logs` (general) (backend/models/audit_log.py:9)

General-purpose, includes: - `parent_lookup.rate_limited` — every brute-force trip - `district.provisioned` — Stripe webhook auto-provision (one row per new district, with welcome_link URL in details) - `superadmin.audit_read` — superadmin reads of any FERPA log - `support.contact` — public support form intake - `pinnova_inbound.fallback` — when leads.db write fails

6. Defense in Depth

6.1 PII never in operational logs

`backend/pii_log_filter.py` attaches a `logging.Filter` to the root + Flask app loggers when `MULTI_TENANT_V2=true` OR `PINNOVA_PII_SCRUB=true`.

Patterns redacted before any handler emits:

PATTERN	REPLACEMENT
Email (`a@b.c`)	`<email>`
Phone (NANP-ish)	`<phone>`
Student legacy IDs (`S-HS-*` `MOCK-STU-*` `H-STU-*` `C-STU-*`)	`<student_id>`
`dob: YYYY-MM-DD` / `date_of_birth: YYYY-MM-DD`	`<dob>`
Bare `YYYY-MM-DD` in PII-context lines	`<date>`
`home_address=...` / `address=...` kv pairs	`<address>`
`name=...` / `parent_name=...` kv pairs	`<name>`

Kept intact: `district_id`, HTTP path, status code, timing, request id, non-PII timestamps (e.g. `bus.last_update`).

6.2 Tenant isolation tested on every CI run

`backend/tests/test_tenant_isolation.py` — 11 tests that provision two districts with **deliberately overlapping student legacy IDs** and verify zero cross-read across every tenant-scoped endpoint. Block 2 commit `a50c1f0`. If a regression introduces cross-tenant leakage, CI catches it before deploy.

6.3 Brute-force protection coverage

`backend/rate_limit.py` — sliding-window counters. Tested in `backend/tests/test_ferpa_hardening.py` (12 tests, block 4 commit `cf6c3b7`). Verified the limit

trips at the 6th request both for successful AND failed lookups — successful counts matter because a legitimate parent doesn't make 5 lookups per minute.

6.4 Rate-limit storage caveat

Sliding-window counters are **process-local** today (in-memory deques). Single-replica DO deploys are fine. When we horizontally scale, swap `check_and_record()` for a Redis-backed implementation behind the same interface — module isolation makes this a single-file change.

7. Retention

CLASS	RETENTION	TRIGGER	BACKED BY
Operational data (students, buses, routes, dispatches, incidents)	30 days after subscription end	District cancels + month-end	Cron (TODO — currently manual)
FERPA audit log (`ferpa_audit_logs`)	**7 years separately**	Survives subscription cancellation	Manual today; automated retention sweep is a §8 followup
General audit log	2 years		Same as above
Stripe billing	Per Stripe's own policy		Out of Pinnova scope
Backups	7 days, then DO-purged	Automated	DO Managed PG

FERPA §99.32 records of access: retained for 7 years per industry common practice (FERPA itself doesn't mandate a specific number — we chose 7 years to mirror tax-record retention so districts can correlate audit trails across systems).

8. Open Followups (not yet shipped — track here)

These are NOT FERPA blockers for go-live but are queued before contracting with the first paying district:

1. **Automated retention sweep cron** — currently manual. A nightly job should mark records `to_delete=true` after the retention class threshold, then run a hard delete weekly with a confirmation lock.
2. **Per-district FERPA dashboard tile** — surface `ferpa_audit_logs` counts (today, week, month) in the admin overview so districts see their own access volume at a glance.
3. **Rate-limit storage → Redis** — for horizontal scale (see §6.4).
4. **Incident `subject_student` audit row** — currently the `/dispatcher/incidents` endpoint doesn't write a FerpaAuditLog row when an incident's `subject_student` is non-null. Add via `_ferpa_audit('student', incident.subject_student, 'read')` in the list response loop.
5. **Pen-test against parent portal** — third-party assessment before first paying district. Targeted at the rate limiter + tenant isolation.
6. **Annual privacy review** — calendar reminder, regenerate this doc.

9. Sub-processors

VENDOR	PURPOSE	DATA SHARED	COMPLIANCE POSTURE
DigitalOcean	Hosting + Managed Postgres	All tenant data (TDE-encrypted)	SOC 2 Type II, ISO 27001
Stripe	Subscription billing	Email + district name + bus count	PCI DSS Level 1
Resend	Welcome / magic-link email	Admin email + district name	SOC 2 Type II (in progress)
Mapbox / OSRM	Map tiles + routing	Coordinates only — never PII	n/a — non-PII
Cloudflare	TLS termination + edge cache	All HTTP traffic in transit	SOC 2 Type II, ISO 27001

Pinnova does not sell or share district data. Sub-processors above are the entire third-party surface.

10. Breach Response

STEP	OWNER	SLA
Detection	On-call (alert via Phantom)	continuous
Triage + scope	JP / Privacy Officer	within 4h of detection
District notification	Privacy Officer → district contact	within **72h** of confirmed breach (state-law minimum is typically 72h; we honor the strictest)
Postmortem	Engineering	within 14 days
Customer-facing incident page	Engineering	within 30 days

Notify endpoints: - `security@pinnovatms.com` — external reporting - `privacy@pinnovatms.com` — district inquiries

11. Auditor checklist (printable)

- [ ] Verify §2 field inventory matches current schema (grep `models/`)
- [ ] Verify §3 access-point list matches current routes (grep `tenant_query(Student|Parent)` in `routes/`)
- [ ] Verify §4 TDE-on by checking DO cluster page
- [ ] Verify §5.1 schema matches `models/audit_log.py:FerpaAuditLog`
- [ ] Run `backend/tests/test_tenant_isolation.py` and confirm 11/11 pass
- [ ] Run `backend/tests/test_ferpa_hardening.py` and confirm 12/12 pass
- [ ] Inspect `pii_log_filter.py` and confirm it's installed in `app.py`
- [ ] Confirm `_PUBLIC_PREFIXES` does NOT include `/api/tenant/*` routes

End of evidence pack. Latest version always available at [`/Volumes/JENA/Playbooks/specs/pinnova_ferpa_audit.md`].