EUDR (EU Deforestation Regulation) Requirements for AgriBackup

Last updated: 2025-10-19

Purpose

This document inventories the project's current APIs and dependencies (backend and frontend), summarizes gaps against EUDR expectations, and provides a prioritized, concrete requirements list (data model, endpoints, security, operational) needed so exporters using AgriBackup can satisfy EUDR due diligence and traceability requirements.

Scope

- Applies to exporter-facing functionality and backend services in `farmers-portal-apis` and the supporting frontend flows in `farmer-portal-frontend`.
- Focused on technical and process requirements necessary to: capture provenance, store evidentiary documents, provide auditable records, enable regulator access/dossiers, and compute basic risk/mitigation.

Summary of current capabilities (what exists today)

- Authentication & RBAC: JWT-based auth and role concepts (FARMER, EXPORTER, SYSTEM_ADMIN, ZONE_SUPERVISOR).
- Exporter onboarding and verification flow: exporters can register without licenseId; can submit license ID and upload a license document via `POST /api/exporters-service/exporter/submit-license-document` and `POST /api/exporters-service/exporter/submit-license`.
- S3 integration: `S3Service` uploads files (produce images and license-documents) and returns public HTTPS URLs. `S3Service` currently uses AWS SDK S3Client and PutObject/ DeleteObject.
- Admin license review: `AdminLicenseService` retrieves exporters with verification status UNDER_REVIEW and exposes review endpoints that change exporter verification status and send templated email notifications.

- Zones: `Zone` entity stores center latitude/longitude and radius (circle-based zone model), with exporter -> zones relationship.
- Some frontend UI elements reference EUDR/traceability and the platform advertises support for secure uploads and presigned URLs in the landing page, but actual presigned URL generation endpoint is not present in S3Service code (S3Service returns public URLs from PutObject).

High-level gaps vs EUDR requirements

The system has a strong foundation (uploads, admin review, exporter entity), but to meet EUDR due diligence the system must implement additional capabilities in the categories below.

1. Data & provenance (critical)

- Missing batch-level traceability model. EUDR requires traceability at lot/batch level (batch UUIDs linking product to parcel and harvest).
- Parcel geometry precision: current Zone model uses circle (center + radius). EUDR expects production unit geometry (plot polygon / GeoJSON) and CRS handling.
- Missing structured document metadata: uploaded documents are stored as URLs only (no structured issuer, issue_date, expiry_date, doc_type, checksum, uploader_id, geotag).

2. Document access & tamper-resistance (critical)

- No presigned URL generator for short-lived access to private S3 objects (S3Service returns public object URLs after PutObject). We need private storage + presigned access.
- No checksum or liveness verification stored for documents.
- No immutable audit trail for document uploads and access events.

3. Risk assessment & mitigation (high)

- No automated risk scoring (country/commodity/producer-based) or integration with external forest monitoring datasets (e.g., Global Forest Watch, Hansen/GLAD/Sentinel alerts).
- No mitigation workflow (requests for additional evidence, on-site checks, conditional holds) surfaced programmatically.

4. Verifier & auditor access (high)

- No dedicated auditor/verifier roles or endpoints to package dossiers for competent authorities.
- No dossier/report endpoint that bundles presigned URLs and machine-readable metadata for a batch/exporter.

5. Storage, retention, and compliance (required)

- No retention policy, scheduled archival, or defined retention period metadata (EUDR expects records to be kept for a number of years — implement >= 5 years by default).
- S3 bucket usage: server-side encryption and strict IAM/bucket policy not enforced in codebase; presigned URL expiry needs to be short and configurable.

6. Spatial monitoring & UI (recommended)

- Polygon capture, map editor/draw UI, and a parcel dashboard with alert overlays are not implemented.

7. Legal & operational (required)

- Supplier questionnaires, mandatory supplier declarations, and standard mitigation templates are not present.

Concrete technical requirements (prioritized)

The following are actionable requirements grouped by priority, each including acceptance criteria.

Priority: Critical (implement ASAP)

1. Private S3 storage + presigned URL generator
    - Change: Store uploaded documents and images in a private S3 bucket (do not return public URLs). Update `S3Service` to:
        - Upload objects with server-side encryption (SSE-S3 or SSE-KMS).
        - Return an internal S3 key (not a public URL) and store that in DB.
        - Implement `generatePresignedUrl(key, expirySeconds)` using AWS SDK S3Presigner with configurable default expiry (e.g., 300s).
    - API: `GET /api/documents/{documentId}/presigned?expirySeconds=300` (authz: auditors/admins/exporter-if-owner).
    - Acceptance: Presigned URLs are short-lived, use HTTPS, and the bucket is private; frontends must fetch content with presigned URL.
2. Documents table and metadata
    - DB: create `documents` table with fields: id (uuid), owner_id, batch_id (nullable), entity_type, doc_type (enum), issuer, issue_date, expiry_date, s3_key, file_name, mime_type, checksum (sha256), geotag (lat/long optional), exif_present (bool), uploader_id, uploaded_at, visibility, tags(json).
    - Backend: new service `DocumentService` to validate file types/sizes, extract EXIF geotags, compute checksum, store metadata, call `S3Service.uploadPrivateFile()`.

- Acceptance: Every uploaded document has a DB record with checksum and uploader metadata.

3. Batch-level traceability model
   - DB: `produce_batches` (or `batch`) table: id (uuid), batch_code (human-readable), produce_listing_id, farm_id, parcel_geojson (or parcel_id foreign key), harvest_date, quantity, unit, created_at, created_by.
   - API: `POST /api/produce-service/batches` to create/assign batch; `GET /api/produce-service/batches/{id}` returns full provenance.
   - Acceptance: Every sale or export line item is tied to a batchId linking to parcel & harvest metadata.
4. Audit log & tamper-evidence
   - DB: `audit_logs` table (append-only): id, entity_type, entity_id, action, actor_id, actor_role, timestamp, details (json), record_hash.
   - Optionally implement periodic hashing of recent changes and store root hash for tamper-evidence (not necessarily blockchain initially).
   - Acceptance: Document uploads, edits, presigned URL generations, and license reviews are logged with actor & timestamp.

Priority: High
5. Dossier / regulator report endpoint

- API: `GET /api/eudr/report?batchId={id}` returns a signed package (JSON + presigned URLs) or a ZIP containing:
  - Batch metadata (producer, parcel geometry, harvest_date, quantity), document metadata and presigned links, audit log entries for the batch.
- AuthZ: Only auditor/admin roles or exporter owner (for their own batches) may call; record access in audit_logs.
- Acceptance: Generated dossier contains all necessary documents and metadata and is accessible via short-lived links.

6. Risk Assessment service & mitigation workflows
   - Implement `RiskAssessmentService` that computes a risk level (LOW/MEDIUM/HIGH) using heuristics: country risk, commodity, land-use change alerts (GFW), supplier history.
   - API: `POST /api/eudr/assess?batchId={id}` or automatic event-based assessment when batch is created or parcel flagged.
   - Add DB fields to store `risk_score` and `risk_reason` and a `mitigation_actions` queue for admins.
   - Acceptance: Risk assessment returns consistent scores, and MEDIUM/HIGH produce triggers a mitigation workflow (manual review, request docs, hold batch).

7. Parcel geometry & spatial database
    - DB: switch to Postgres + PostGIS for spatial queries, or add GeoJSON columns and a PostGIS migration.
    - Store parcel polygons (GeoJSON) for farms/plots, not just a radius circle. Provide a shape editor in frontend zone/farm UI (map draw poly).
    - Acceptance: Parcels can be queried for intersection with forest-loss alerts and used in risk scoring.

Priority: Medium
8. Third-party verifier roles and certificate handling

- Add Verifier role and API for verifier uploads/notes. Allow linking third-party certificate metadata to batches.
- Acceptance: Verifier users can upload verification reports and those are reflected in the batch dossier.

9. External data integrations
    - Integrate Global Forest Watch alerts, Sentinel/Hansen datasets or a third-party provider to flag land-cover change for a parcel.
    - Acceptance: The system receives alerts and flags parcels; these feed the risk assessment.
10. Retention & archival

- Define retention policy (default >= 5 years), implement scheduled archival job to move older data to cold storage and/or mark as archived; maintain audit trail entries.
- Acceptance: Documents older than retention are archived and still retrievable by regulators per policy.

Security and infrastructure changes

- AWS S3
    - Use private buckets and server-side encryption (SSE-S3 or SSE-KMS).
    - Generate presigned URLs with S3Presigner; configure short expiration defaults (e.g., 5 minutes) and allow override per endpoint with max cap.
    - Use least-privilege IAM roles for the app (restrict PutObject/GetObject/DeleteObject to specific prefixes).
- Database & storage
    - Prefer Postgres + PostGIS for spatial queries and indexing, or add spatial support via hstore/JSONB if PostGIS is not an option.
    - Use SHA-256 (or stronger) for file checksums and store them in the `documents` table.

- Encryption & privacy
  - Ensure HTTPS for all endpoints; confirm mailers/notifications do not leak confidential data.
  - Assess GDPR impact for personal data in audit logs and documents; add consent capture and data processing notes in the UI and privacy policy.

API and DB change summary (concrete)

DB Migrations (Liquibase changeSets):

- Add table `documents` (fields described above).
- Add table `produce_batches`.
- Add table `audit_logs`.
- Add columns to `zones` or `farms` for `parcel_geojson` (nullable) and `geo_precision`.

New/Changed backend endpoints (examples)

- POST /api/documents (multipart + metadata) -> create Document record + store object privately
- GET /api/documents/{id} (metadata only)
- GET /api/documents/{id}/presigned?expirySeconds=300
- POST /api/produce-service/batches (create batch, link to parcel & docs)
- GET /api/produce-service/batches/{id}
- GET /api/eudr/report?batchId={id} (generate packaged dossier ZIP or JSON + presigned links)
- POST /api/eudr/assess?batchId={id} (run risk assessment)
- GET /api/eudr/alerts?parcelId={id}

Dependencies & libraries to add or upgrade

- AWS SDK: add S3Presigner usage (software.amazon.awssdk:s3) — already using AWS SDK v2; ensure S3Presigner is available and tested.
- Postgres + PostGIS support in backend: add `org.postgresql:postgresql` driver and consider `com.vladmihalcea:hibernate-types-52` for JSON/GeoJSON mapping.
- Add geospatial client libs on frontend: `leaflet` / `vue2-leaflet` is already present; add drawing plugin (leaflet-draw) for polygon capture.
- Add a scheduler (Spring @Scheduled) or Quartz for archival/retention jobs if not present.

Operational & process requirements

- Operational SOPs for EUDR: define who in the organization acts on high-risk findings, timelines for mitigation, and how to notify competent authorities.
- Training for admin/verifier roles to interpret risk scores and review evidence.
- Logging & monitoring: ensure access logs, audit logs, and alerts for suspicious behavior; connect to a log aggregator and alerting system.

Acceptance criteria (per feature)

- Documents: uploaded docs must be stored privately with checksum, metadata, and be retrievable via presigned URL; tests should verify checksum and that a generated presigned URL allows GET only for its expiry window.
- Batches: creating a batch links to a parcel geojson and harvest date; retrieving a batch returns full chain-of-custody.
- Dossier: calling the dossier endpoint returns a ZIP (or machine-readable JSON) containing metadata, presigned links, and audit trail entries, and calling it generates audit logs for access.
- Risk: risk assessment returns a score and a textual rationale. MEDIUM/HIGH must create an actionable mitigation entry in a review queue.

Implementation roadmap & estimates (rough)

Phase 1 (2-4 weeks) — Foundation

- Implement Documents table + `DocumentService` and S3 private uploads + presigned URL generator. Add endpoint for presigned links.
- Add `produce_batches` table and batch creation endpoint.

Phase 2 (3-6 weeks)

- Implement audit_logs and presigned access logging.
- Implement EUDR dossier generation endpoint and exporter/admin UI to download dossier.
- Add retention/archival scheduler and initial retention policy.

Phase 3 (4-8 weeks)

- Add risk assessment service (basic country/commodity heuristics), and integrate an external alert source (Global Forest Watch) for parcel monitoring.
- Add verifier role and advanced mitigation workflow.

Longer-term (6+ weeks)

- PostGIS migration and advanced spatial queries.
- Tamper-evidence cryptographic chain (hashchain) and formal audit signing if required.

Immediate next steps I can take for you

1. Create Liquibase changeSet for the `documents` table and `produce_batches` table (fast, low-risk).
2. Implement `S3Service.generatePresignedUrl(...)` and a new documents controller endpoint for presigned access (small, testable change).
3. Scaffold `DocumentService` to extract EXIF and compute checksums on upload.

If you want me to start coding, tell me which of the immediate next steps (1..3) to implement first and I'll create the necessary files and tests.

Notes & assumptions

- I inspected the backend `S3Service`, `ExporterController`, `AdminLicenseService`, exporter domain and `PROJECT_DOCUMENTATION.md` to build this list. The codebase already supports file uploads but returns public S3 URLs; for compliance we should make storage private and serve via presigned URLs.
- I recommend Postgres+PostGIS for robust spatial queries; an interim approach is to store GeoJSON in JSONB and use external tooling for spatial analysis until PostGIS migration is feasible.
- Legal/regulatory interpretation is context-dependent; this document lists technical and operational capabilities that implement typical EUDR due-diligence needs but does not substitute for legal counsel.

Contact

If you'd like, I can implement the presigned URL endpoint and documents table now (low-risk). Which immediate step should I start with?