Aadhaar eSign OTP Service – Architecture & Implementation v1.0 (Node.js)

Author: Sourabh A.
Date: 21 Aug 2025
Status: Draft for Review

---

0. Executive Summary

This document details a production-grade, OTP-based Aadhaar eSign service acting as an Aggregation Service Provider (ASP) bridge between client applications and the CVL eSign Gateway. It covers the end-to-end architecture, data flows, API contracts, security model, deployment topology, observability, and operational runbooks. All artifacts are designed for scale, security, and auditability.

---

1. Goal

Provide a secure, scalable Node.js service to orchestrate Aadhaar eSign flows via CVL (OTP/eKYC). - Offer stable, versioned REST APIs and reliable webhooks to client applications. - Ensure strong security (at rest & in transit), audit logging, and compliance-friendly records. - Support PDF signing (PKCS#7/CAdES) and XML signing per CVL/eSign specs.

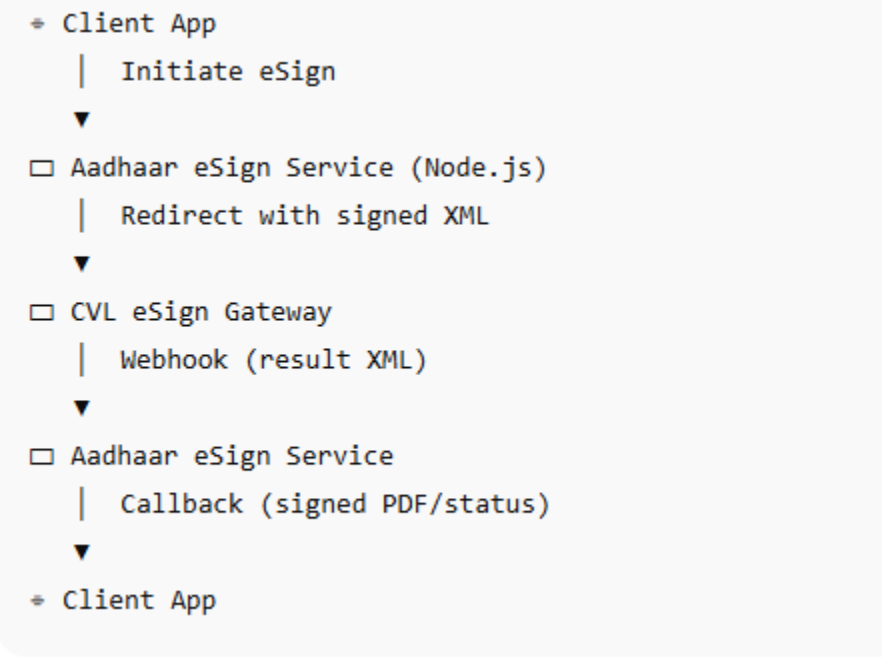
---

2. High-Level Architecture

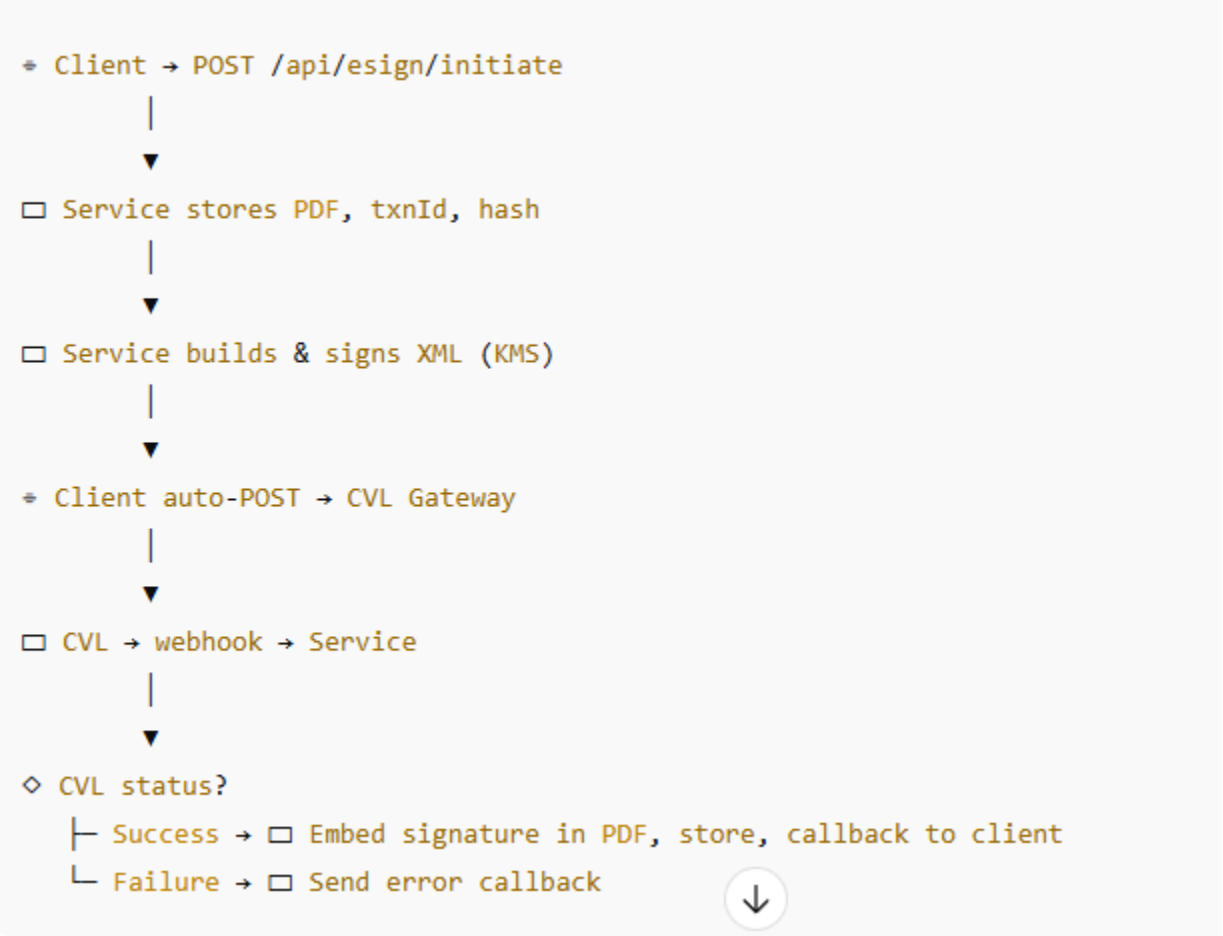
2.1 Components

- **Client Application** — Initiates eSign, receives redirect HTML and final signed document via webhook.
- **Aadhaar eSign Service (Node.js/Express)** — API layer, XML signing, PDF signing, persistence, webhook orchestration.
- **CVL eSign Gateway** — Performs Aadhaar eKYC and eSign; posts result to our webhook endpoint.
- **Relational DB (MySQL)** — ACID store for transactions, responses, signed docs, audit logs, clients.
- **Object Storage (S3)** — Encrypted storage for PDFs and artifacts.
- **KMS/HSM** — Secure private key storage & signing operations.
- **Message Queue (optional)** — For async PDF signing and webhook retries (e.g., SQS/RabbitMQ/Redis Streams).

2.2 Context Flow (Simple Inline Diagram)



2.3 End-to-End Sequence



---

3. APIs (v1)

3.1 Authentication

- Header `X-API-Key: <key>` or mTLS. Keys are per client in **esign_clients**.
- Idempotency header `Idempotency-Key` supported for POST `/initiate`.

3.2 Endpoints

POST /api/esign/initiate - **Input (multipart/form-data):** - `pdf` (file, required) - `signerName` (string) - `signCoordinates` (json: `{ page, x, y, width, height }`) - `clientWebhookUrl` (string, https) - `clientId` (string) - `metadata` (json) - **Responses:** - `200`: `{ txnId, redirectHtml }` - `4xx/5xx`: error JSON (see Error Catalog)

POST /api/esign/webhook (CVL → Service) - **Body:** XML (per CVL). Includes txnId, status, signature block. - **Response:** `204 No Content` if accepted.

POST /api/esign/callback (Service → Client webhook) - **Body (JSON):**

```
{
  "txnId": "string",
  "signStatus": "success|failed",
  "signedFile": "<base64>|<https link>",
  "esignResponseXml": "<xml>",
  "originalRequest": { },
  "errorMessage": "string|null"
}
```

---

4. Error Handling & Observability

4.1 Error Catalog (examples)

Code	HTTP	Message	Action
ESIGN-001	400	Invalid input/schema	Fix request payload
ESIGN-002	401	Invalid API key	Regenerate key
ESIGN-010	422	Unsupported PDF	Re-export PDF
ESIGN-020	424	CVL signature verify failed	Investigate certificates
ESIGN-030	500	XML signing failure	Check KMS/HSM
ESIGN-040	504	Callback timeout	Will retry

---

5. Implementation Plan & Milestones

1. **Contracts & Schemas**: Freeze API v1 and JSON Schemas.
2. **Initiate Endpoint**: Uploads, hashing, XML builder, redirect HTML.
3. **Webhook Endpoint**: Verify CVL XML signature, persist response.
4. **PDF Signing Worker**: PKCS#7 signing, storage, link generation.
5. **E2E in CVL Sandbox**: Test matrices, error cases, rate/scale.

---

6. Tech Stack & Libraries

- **Runtime:** Node.js LTS, Express.js
- **PDF:** `pdf-lib`, `node-signpdf` (CAdES), or external Java/.NET helper via gRPC/HTTP
- **XML:** `xmlbuilder2`, `xml-crypto`, `xml-c14n`
- **Crypto:** Node `crypto`, KMS SDK, `pkcs11js` (if HSM)
- **DB/ORM:** MySQL + Sequelize/TypeORM
- **Storage:** S3
- **Queues:** SQS/RabbitMQ/Redis Streams

---

7. Security Controls – Detailed

- Input validation & JSON schema enforcement on all APIs.
- Strict Content-Security-Policy for redirect HTML (auto-post FORM only).
- Signature and certificate chain verification for CVL webhook.
