

EUDR Compliance Certificate NFT Lifecycle & State Management

Overview

The EUDR Compliance Certificate NFT system provides an immutable, blockchain-based proof of regulatory compliance for agricultural shipments entering the EU. This document details the complete lifecycle, state transitions, and technical implementation.

Table of Contents

1. [Core Concept](#)
2. [Supply Chain Actors & Account Creation](#)
3. [Certificate Lifecycle States](#)
4. [Complete Flow: Farm to Customs](#)
5. [State Transitions & Triggers](#)
6. [Technical Implementation](#)
7. [Blockchain Recording Strategy](#)
8. [Error Handling & Edge Cases](#)

Core Concept

What is the EUDR Compliance Certificate NFT?

Purpose: Mandatory proof that a shipment meets EU Deforestation Regulation (EUDR) requirements

Nature:

- Non-Fungible Token (NFT) - each certificate is unique
- **NOT an incentive or reward** - it's a regulatory requirement
- Acts as a digital passport for the shipment
- Transferable with physical goods ownership
- Immutable record on Hedera blockchain

Key Principle:

"Like a passport for travel - compliance is mandatory, the certificate is proof"

Certificate Properties

Token Name: EUDR Compliance Certificate
Token Symbol: EUDR-CERT
Token Type: NFT (Non-Fungible)
Blockchain: Hedera Hashgraph
Supply: Unlimited (one per compliant shipment)
Transferable: Yes (with shipment ownership)
Revocable: Yes (via freeze mechanism)

Supply Chain Actors & Account Creation

Actor Types & Their Hedera Accounts

1. Farmer (Optional - Future)

- **Current Status:** No Hedera account (tracked via GPS coordinates only)
- **Future Consideration:** Individual farmer accounts for carbon credit distribution
- **Decision Status:** Deferred pending use case validation

2. Aggregator

- **Role:** Collects produce from multiple farmers
- **Account Creation:** Automatic during registration
- **Initial Balance:** 10 HBAR
- **NFT Association:** Automatic (EUDR Certificate NFT)
- **Database:** aggregators.hedera_account_id
- **Credentials:** Encrypted in hedera_account_credentials table

```
// Aggregator Registration Flow
1. User submits registration form
2. System creates UserProfile
3. System creates Hedera account (10 HBAR)
4. Private key encrypted with AES-256-GCM
5. Aggregator entity created with hedera_account_id
6. Credentials stored in hedera_account_credentials
7. EUDR Certificate NFT automatically associated
8. Account creation recorded on HCS
9. Token association recorded on HCS
```

3. Processor

- **Role:** Processes raw agricultural products
- **Account Creation:** Automatic during registration
- **Initial Balance:** 10 HBAR
- **NFT Association:** Automatic (EUDR Certificate NFT)

- **Database:** processors.hedera_account_id
- **Credentials:** Encrypted in hedera_account_credentials table

```
// Processor Registration Flow
1. User submits facility registration
2. System creates UserProfile
3. System creates Hedera account (10 HBAR)
4. Private key encrypted with AES-256-GCM
5. Processor entity created with hedera_account_id
6. Credentials stored in hedera_account_credentials
7. EUDR Certificate NFT automatically associated
8. Account creation recorded on HCS
9. Token association recorded on HCS
```

4. Exporter ⚠️

- **Role:** Ships products internationally
- **Account Creation:** Manual (no standard registration flow)
- **Current Status:** Hedera integration exists but not auto-created
- **Database:** exporters.hedera_account_id (exists but populated manually)
- **Note:** Can be added when exporter registration API is implemented

5. Importer ✅

- **Role:** Receives international shipments
- **Account Creation:** Automatic during registration
- **Initial Balance:** 10 HBAR
- **NFT Association:** Automatic (EUDR Certificate NFT)
- **Database:** importers.hedera_account_id
- **Credentials:** Encrypted in hedera_account_credentials table

```
// Importer Registration Flow
1. User submits company registration
2. System creates UserProfile
3. System creates Hedera account (10 HBAR)
4. Private key encrypted with AES-256-GCM
5. Importer entity created with hedera_account_id
6. Credentials stored in hedera_account_credentials
7. EUDR Certificate NFT automatically associated
8. Importer creation recorded on HCS
9. Token association recorded on HCS
```

Certificate Lifecycle States

State 1: NOT_CREATED

Description: Certificate doesn't exist yet

Conditions:

- Shipment is being assembled
- Traceability data is being collected
- EUDR checks have not been completed

Shipment Status: PENDING OR IN_PREPARATION





Certificate NFT: Does not exist

Blockchain State: No certificate record

State 2: **PENDING_VERIFICATION**

Description: Data collection complete, awaiting EUDR verification

Conditions:

- All traceability data collected:
 -  Farmer GPS coordinates captured
 -  Production unit locations recorded
 -  Supply chain actors identified (Aggregator → Processor → Exporter)
 -  Product quantities and dates logged
- EUDR compliance checks initiated

Shipment Status: UNDER_REVIEW

EUDR Compliance Status: PENDING_VERIFICATION

Certificate NFT: Not yet minted

Blockchain State: Supply chain events recorded on HCS



EUDR Checks Performed:

1. Origin Country Risk Assessment
 - High Risk: Brazil, Indonesia, DRC, etc.
 - Medium Risk: Other tropical countries
 - Low Risk: EU member states
2. Deforestation Verification
 - GPS coordinates analyzed against deforestation maps
 - Satellite imagery cross-referenced
 - Historical land use checked
3. Traceability Completeness
 - All supply chain actors verified
 - Quantities match at each stage
 - Dates are chronologically consistent
 - No gaps in custody chain
4. Due Diligence Statement
 - Risk assessment documented
 - Mitigation measures recorded
 - Operator declarations validated

State 3: **COMPLIANT** (Certificate Issued)

Description: Shipment passed all checks, certificate NFT minted and issued

Conditions:

-  Deforestation-free verification passed
-  Complete traceability chain validated
-  GPS coordinates verified
-  Risk assessment completed and accepted
-  Due diligence statement generated

Shipment Status: APPROVED

EUDR Compliance Status: COMPLIANT

Certificate NFT: ISSUED to exporter's Hedera account

Blockchain State:

- Certificate NFT minted (serial number: 1 per shipment)
- NFT transferred to exporter's account
- Issuance recorded on HCS with full compliance data

Certificate Metadata:

```
{
  "shipmentId": "SHP-2025-001",
  "originCountry": "Kenya",
  "riskLevel": "LOW",
  "totalFarmers": 45,
  "totalProductionUnits": 67,
  "gpsCoordinatesCount": 67,
  "deforestationStatus": "VERIFIED_FREE",
  "traceabilityHash": "0x7d3f...",
  "issuedAt": "2025-10-25T10:30:00Z",
  "issuedTo": "0.0.123456", // Exporter's Hedera account
  "nftSerialNumber": 1
}
```

Technical Implementation:

```
// In HederaTokenService
fun issueComplianceCertificateNft(
    shipmentId: String,
    exporterAccountId: AccountId,
    complianceData: Map<String, String>
): String {
    // 1. Mint ONE unique NFT for this shipment
    val transaction = TokenMintTransaction()
        .setTokenId(eudrComplianceCertificateNftId)
        .setAmount(1) // ONE NFT per shipment
        .execute(client)

    // 2. Transfer NFT to exporter
    transferCertificateNft(tokenId, exporterAccountId, 1)

    // 3. Record on HCS
    hederaConsensusService.recordComplianceCertificateIssuance(
        shipmentId = shipmentId,
        exporterAccountId = exporterAccountId,
        nftSerialNumber = 1,
        complianceData = complianceData
    )

    return transactionId
}
```

State 4: **IN_TRANSIT** (Certificate Held by Exporter)

Description: Shipment exported, certificate travels with it

Conditions:

- Shipment physically in transit

- Bill of lading issued
- Certificate NFT in exporter's account

Shipment Status: IN_TRANSIT

EUDR Compliance Status: COMPLIANT

Certificate NFT: In **exporter's Hedera account**

Certificate Balance: Exporter = 1 NFT

Key Activities:

- Exporter maintains custody of both physical goods and digital certificate
- Certificate can be verified by customs in advance
- Blockchain provides real-time verification capability

State 5: TRANSFERRED_TO_IMPORTER (Certificate Transferred)

Description: Shipment ownership transferred, certificate follows

Conditions:

- Importer accepts shipment
- Ownership legally transferred
- Certificate NFT transferred to importer's account

Shipment Status: CUSTOMS_CLEARANCE OR QUALITY_INSPECTION

EUDR Compliance Status: COMPLIANT

Certificate NFT: In **importer's Hedera account**

Certificate Balance:

- Exporter = 0 NFT (transferred out)
- Importer = 1 NFT (received)

Technical Implementation:

```
// Certificate Transfer
fun transferComplianceCertificateNft(
    fromAccountId: AccountId,    // Exporter
    toAccountId: AccountId,      // Importer
    shipmentId: String
): Boolean {
    // 1. Transfer NFT on Hedera
    val transaction = TransferTransaction()
        .addTokenTransfer(tokenId, fromAccountId, -1)
        .addTokenTransfer(tokenId, toAccountId, 1)
        .execute(client)

    // 2. Record transfer on HCS
    hederaConsensusService.recordComplianceCertificateTransfer(
        shipmentId = shipmentId,
        fromAccountId = fromAccountId,
        toAccountId = toAccountId,
        nftSerialNumber = 1
    )

    return true
}
```

Blockchain State Change:

BEFORE:

Exporter Account (0.0.123456): [EUDR-CERT #1]
 Importer Account (0.0.789012): []

TRANSACTION:

TransferTransaction
 - From: 0.0.123456, Amount: -1 EUDR-CERT
 - To: 0.0.789012, Amount: +1 EUDR-CERT

AFTER:

Exporter Account (0.0.123456): []
 Importer Account (0.0.789012): [EUDR-CERT #1]

State 6: **CUSTOMS_VERIFIED** (Certificate Checked)

Description: Customs authority verified certificate authenticity

Conditions:

- Customs officer queries blockchain
- Certificate authenticity confirmed
- Compliance data matches shipment

Shipment Status: CUSTOMS_CLEARANCE

EUDR Compliance Status: COMPLIANT

Certificate NFT: Still in importer's account

Certificate Balance: Importer = 1 NFT

Verification Process:

```
// Customs Verification (Read-Only)
fun verifyComplianceCertificate(
    importerAccountId: AccountId,
    shipmentId: String
): Boolean {
    // 1. Check if importer has the certificate
    val hasNFT = hasValidComplianceCertificate(importerAccountId)

    // 2. Query blockchain for certificate history
    val certificateData = hederaConsensusService
        .queryCertificateHistory(shipmentId)

    // 3. Validate:
    //     - Certificate exists
    //     - Issued for this shipment
    //     - Transferred to correct importer
    //     - Not frozen/revoked

    return hasNFT && certificateData.isValid && !certificateData.isFrozen
}
```

Public Verification:

- Anyone can verify on HashScan: <https://hashscan.io/testnet/token/0.0.xxxxx>
- Transparency ensures trust without intermediaries

State 7: **DELIVERED** (Certificate Retained)

Description: Shipment delivered, certificate archived

Conditions:

- Physical goods delivered to importer
- Quality inspection passed
- Certificate remains as permanent record

Shipment Status: DELIVERED

EUDR Compliance Status: COMPLIANT

Certificate NFT: Remains in **importer's account**

Certificate Purpose: Historical proof of compliance

Long-term Retention:

- Certificate never destroyed (immutable blockchain record)
- Serves as audit trail for 5+ years (EUDR requirement)
- Can be queried for historical compliance audits

State 8: **FROZEN** (Certificate Revoked) ❌

Description: Fraud detected, certificate invalidated

Conditions:

- Fraud investigation concluded
- False GPS data discovered
- Deforestation found on origin land
- Other compliance violations

Shipment Status: REJECTED

EUDR Compliance Status: NON_COMPLIANT

Certificate NFT: **FROZEN** in holder's account

Certificate Transferable: ❌ NO (frozen assets cannot be transferred)

Technical Implementation:

```
// Freeze Certificate (Revocation)
fun freezeComplianceCertificateNft(
    accountId: AccountId,
    reason: String // "FRAUD_DETECTED", "FALSE_GPS_DATA", etc.
): Boolean {
    // 1. Freeze NFT in account
    val transaction = TokenFreezeTransaction()
        .setTokenId(eudrComplianceCertificateNftId)
        .setAccountId(accountId)
        .execute(client)

    // 2. Record freezing on HCS
    hederaConsensusService.recordComplianceCertificateFreeze(
        accountId = accountId,
        nftSerialNumber = 1,
        reason = reason
    )

    return true
}
```

Consequences:

- ❌ Shipment cannot be sold/transferred
- ❌ Customs clearance denied
- ❌ Compliance status permanently marked
- ⚠️ Importer may face penalties

- ⚠️ Exporter may lose certification

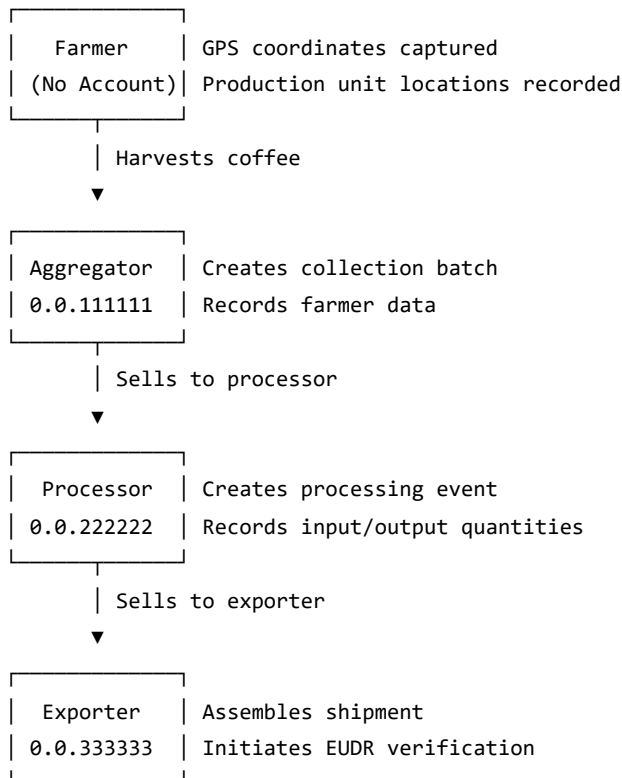
Unfreezing (if investigation was wrong):

```
fun unfreezeComplianceCertificateNft(accountId: AccountId): Boolean {
    // Only platform operator can unfreeze
    // Requires admin approval
    val transaction = TokenUnfreezeTransaction()
        .setTokenId(eudrComplianceCertificateNftId)
        .setAccountId(accountId)
        .execute(client)

    return true
}
```

Complete Flow: Farm to Customs

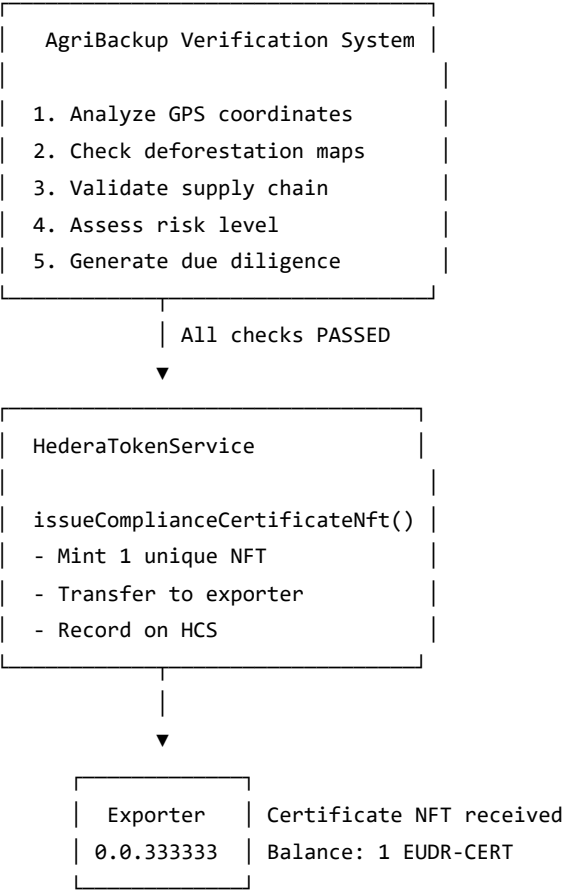
Phase 1: Data Collection (No Certificate Yet)



Blockchain State:

- ✅ HCS: Collection batch recorded
- ✅ HCS: Processing event recorded
- ✅ HCS: Shipment creation recorded
- ❌ NFT: Not yet created

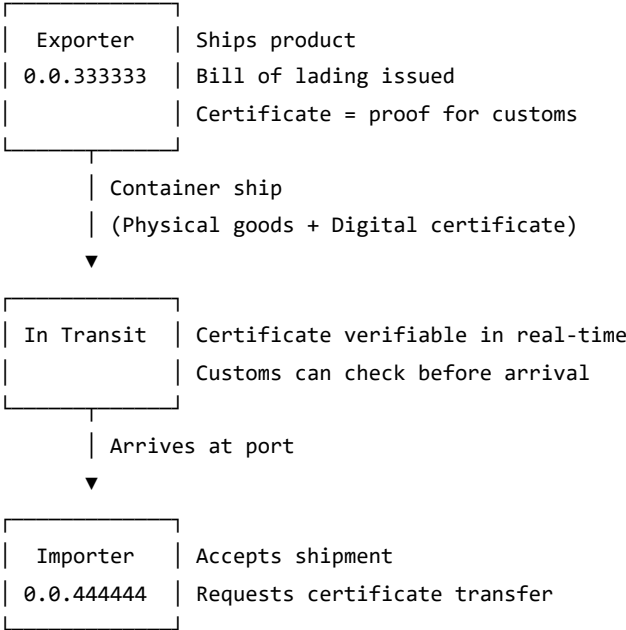
Phase 2: EUDR Verification & Certificate Issuance



Blockchain State:

- ✔ NFT: Minted (serial #1)
- ✔ NFT: In exporter's account
- ✔ HCS: Issuance recorded with compliance data

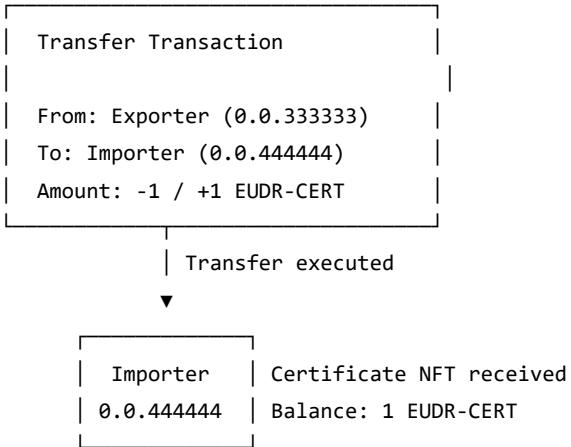
Phase 3: International Shipping



Blockchain State:

- ✔ NFT: Still in exporter's account
- ✔ HCS: In-transit status recorded

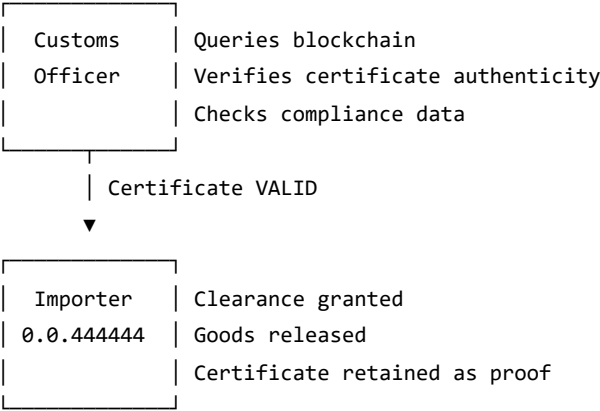
Phase 4: Ownership Transfer



Blockchain State:

- ✔ NFT: Transferred to importer
- ✔ HCS: Transfer recorded
- ✔ Exporter Balance: 0 EUDR-CERT
- ✔ Importer Balance: 1 EUDR-CERT

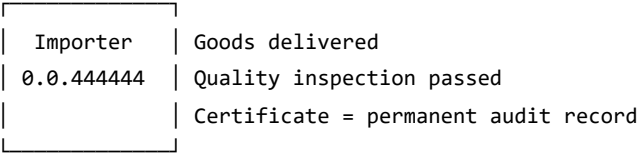
Phase 5: Customs Clearance



Blockchain Query:

- ✔ Certificate exists
- ✔ Issued for this shipment
- ✔ Transferred to correct importer
- ✔ Not frozen/revoked
- ✔ Compliance data matches declaration

Phase 6: Delivery & Archival

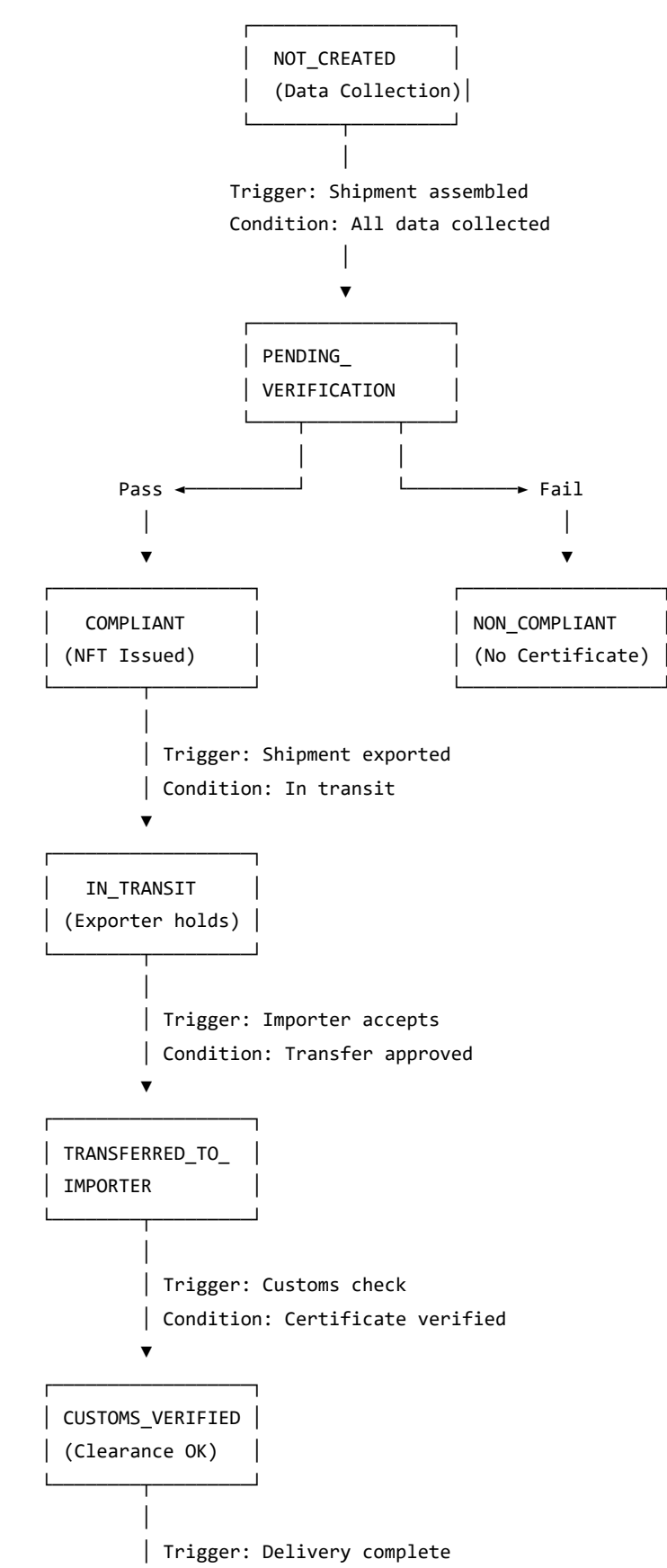


Blockchain State:

- ✔ NFT: Remains in importer's account (permanent)
- ✔ HCS: Delivery recorded
- ✔ Audit Trail: Complete from farm to customs
- ✔ Compliance Proof: Retained for 5+ years

State Transitions & Triggers

State Diagram



| Condition: Quality passed



ANY STATE → Fraud Detected → FROZEN (Revoked)

Trigger Matrix

Current State	Trigger	Conditions Required	Next State	Blockchain Action
NOT_CREATED	Shipment assembled	All traceability data collected	PENDING_VERIFICATION	HCS: Verification initiated
PENDING_VERIFICATION	EUDR checks passed	GPS valid + No deforestation + Complete chain	COMPLIANT	NFT: Minted & Issued; HCS: Recorded
PENDING_VERIFICATION	EUDR checks failed	Missing data OR deforestation found	NON_COMPLIANT	HCS: Failure reason recorded
COMPLIANT	Shipment exported	Bill of lading issued	IN_TRANSIT	HCS: Export recorded
IN_TRANSIT	Importer accepts	Transfer request approved	TRANSFERRED_TO_IMPORTER	NFT: Transferred; HCS: Transfer recorded
TRANSFERRED_TO_IMPORTER	Customs inspection	Certificate validated	CUSTOMS_VERIFIED	HCS: Verification recorded
CUSTOMS_VERIFIED	Delivery confirmed	Quality inspection passed	DELIVERED	HCS: Delivery recorded
ANY	Fraud detected	Investigation completed	FROZEN	NFT: Frozen; HCS:

Current State	Trigger	Conditions Required	Next State	Blockchain Action
				Freeze reason recorded
FROZEN	Investigation cleared	Admin approval	UNFROZEN (previous state)	NFT: Unfrozen; HCS: Unfreeze recorded

Technical Implementation

Database Schema

hedera_account_credentials Table

```
CREATE TABLE hedera_account_credentials (  
  id VARCHAR(36) PRIMARY KEY,  
  user_id VARCHAR(50) NOT NULL,  
  entity_type VARCHAR(50) NOT NULL, -- 'AGGREGATOR', 'PROCESSOR', 'IMPORTER'  
  entity_id VARCHAR(50) NOT NULL,  
  hedera_account_id VARCHAR(50) UNIQUE NOT NULL,  
  public_key VARCHAR(200) NOT NULL,  
  encrypted_private_key VARCHAR(500) NOT NULL, -- AES-256-GCM encrypted  
  creation_transaction_id VARCHAR(100),  
  is_active BOOLEAN DEFAULT TRUE,  
  tokens_associated TEXT, -- JSON array: ["0.0.xxxxx"]  
  created_at DATETIME,  
  last_used_at DATETIME,  
  FOREIGN KEY (user_id) REFERENCES user_profiles(id) ON DELETE CASCADE  
);
```

Updated Entity Tables

```
-- Aggregators
ALTER TABLE aggregators
ADD COLUMN hedera_account_id VARCHAR(50);

-- Processors
ALTER TABLE processors
ADD COLUMN hedera_account_id VARCHAR(50);

-- Importers (already had column)
-- importers.hedera_account_id already exists

-- Import Shipments (for certificate tracking)
ALTER TABLE import_shipments
ADD COLUMN compliance_certificate_nft_id VARCHAR(50),
ADD COLUMN compliance_certificate_serial_number BIGINT,
ADD COLUMN compliance_certificate_transaction_id VARCHAR(100),
ADD COLUMN current_owner_account_id VARCHAR(50);
```

Service Layer Architecture



Key Code Snippets

1. Account Creation Pattern (Applied to all entities)

```
fun createEntity(dto: CreateEntityDto): EntityResponseDto {
    // 1. Create user profile
    val user = UserProfile(...)
    userRepository.save(user)

    // 2. Create Hedera account
    val hederaAccount = try {
        hederaAccountService.createHederaAccount(
            initialBalance = Hbar.from(10),
            memo = "AgriBackup [EntityType]: ${dto.name}"
        )
    } catch (e: Exception) {
        null // Graceful degradation
    }

    // 3. Create entity
    val entity = Entity(
        ...,
        hederaAccountId = hederaAccount?.accountId,
        userProfile = user
    )
    val saved = repository.save(entity)

    // 4. Store encrypted credentials
    if (hederaAccount != null) {
        val credentials = HederaAccountCredentials(
            userId = user.id,
            entityType = "ENTITY_TYPE",
            entityId = saved.id,
            hederaAccountId = hederaAccount.accountId,
            publicKey = hederaAccount.publicKey,
            encryptedPrivateKey = hederaAccount.encryptedPrivateKey,
            tokensAssociated = [],
            ...
        )
        credentialsRepository.save(credentials)

    // 5. Associate with EUDR Certificate NFT
    val nftId = hederaTokenService.getEudrComplianceCertificateNftId()
    if (nftId != null) {
        hederaAccountService.associateTokenWithAccount(
            hederaAccount.accountId,
            hederaAccount.encryptedPrivateKey,
            nftId
        )
        credentials.tokensAssociated = ""["${nftId}"]""
        credentialsRepository.save(credentials)
    }
}
```

```

    }

    return mapToDto(saved)
}

```

2. Certificate Issuance (When shipment passes EUDR)

```

fun verifyAndCertifyShipment(shipmentId: String): ShipmentResponseDto {
    val shipment = getShipment(shipmentId)

    // Run EUDR compliance checks
    val complianceResult = eudrVerificationService.verify(shipment)

    if (complianceResult.isCompliant) {
        // Get exporter's Hedera credentials
        val exporterCredentials = credentialsRepository
            .findByEntityTypeAndEntityId("EXPORTER", shipment.exporterId)
            .orElseThrow()

        // Issue EUDR Compliance Certificate NFT
        val txId = hederaTokenService.issueComplianceCertificateNft(
            shipmentId = shipment.id,
            exporterAccountId = AccountId.fromString(exporterCredentials.hederaAccountId),
            complianceData = mapOf(
                "originCountry" to complianceResult.originCountry,
                "riskLevel" to complianceResult.riskLevel.name,
                "totalFarmers" to complianceResult.farmerCount.toString(),
                "totalProductionUnits" to complianceResult.productionUnitCount.toString(),
                "gpsCoordinatesCount" to complianceResult.gpsCount.toString(),
                "deforestationStatus" to "VERIFIED_FREE",
                "traceabilityHash" to complianceResult.traceabilityHash
            )
        )

        // Update shipment with certificate info
        shipment.complianceCertificateNftId = hederaTokenService.getEudrComplianceCertificateNftId().toString()
        shipment.complianceCertificateSerialNumber = 1
        shipment.complianceCertificateTransactionId = txId
        shipment.eudrComplianceStatus = EudrComplianceStatus.COMPLIANT
        shipmentRepository.save(shipment)
    } else {
        shipment.eudrComplianceStatus = EudrComplianceStatus.NON_COMPLIANT
        shipmentRepository.save(shipment)
    }

    return mapToDto(shipment)
}

```

3. Certificate Transfer (When importer accepts shipment)

```
fun transferShipmentToImporter(shipmentId: String, importerId: String) {  
    val shipment = getShipment(shipmentId)  
  
    // Get credentials for both parties  
    val exporterCreds = credentialsRepository  
        .findByEntityTypeAndEntityId("EXPORTER", shipment.exporterId)  
        .orElseThrow()  
  
    val importerCreds = credentialsRepository  
        .findByEntityTypeAndEntityId("IMPORTER", importerId)  
        .orElseThrow()  
  
    // Transfer EUDR Certificate NFT  
    val success = hederaTokenService.transferComplianceCertificateNft(  
        fromAccountId = AccountId.fromString(exporterCreds.hederaAccountId),  
        toAccountId = AccountId.fromString(importerCreds.hederaAccountId),  
        shipmentId = shipmentId  
    )  
  
    if (success) {  
        shipment.currentOwnerAccountId = importerCreds.hederaAccountId  
        shipment.shipmentStatus = ShipmentStatus.CUSTOMS_CLEARANCE  
        shipmentRepository.save(shipment)  
    }  
}
```

4. Certificate Verification (Customs check)

```
fun verifyCustomsCompliance(shipmentId: String): CustomsVerificationResult {  
    val shipment = getShipment(shipmentId)  
  
    // Get importer's credentials  
    val importerCreds = credentialsRepository  
        .findByEntityTypeAndEntityId("IMPORTER", shipment.importerId)  
        .orElseThrow()  
  
    val importerAccount = AccountId.fromString(importerCreds.hederaAccountId)  
  
    // Check if importer has the certificate NFT  
    val hasNFT = hederaTokenService.hasValidComplianceCertificate(importerAccount)  
  
    if (hasNFT) {  
        // Query blockchain for certificate history  
        val certificateHistory = hederaConsensusService.queryCertificateHistory(shipmentId)  
  
        // Validate certificate data  
        val isValid = certificateHistory.issuedFor == shipmentId &&  
            certificateHistory.currentOwner == importerAccount.toString() &&  
            !certificateHistory.isFrozen  
  
        if (isValid) {  
            shipment.shipmentStatus = ShipmentStatus.APPROVED  
            shipmentRepository.save(shipment)  
  
            return CustomsVerificationResult(  
                approved = true,  
                certificateValid = true,  
                complianceStatus = "COMPLIANT",  
                message = "EUDR certificate verified - clearance granted"  
            )  
        }  
    }  
  
    return CustomsVerificationResult(  
        approved = false,  
        certificateValid = false,  
        complianceStatus = "NON_COMPLIANT",  
        message = "No valid EUDR certificate found"  
    )  
}
```

Blockchain Recording Strategy

Hedera Consensus Service (HCS) Events

All state changes are recorded on HCS for transparency and auditability:

Event Types

Event Type	When	Data Recorded
HEDERA_ACCOUNT_CREATED	Entity registration	Account ID, entity type, memo
TOKEN_ASSOCIATED	NFT association	Account ID, token ID
EUDR_CERTIFICATE_ISSUED	Shipment compliance verified	Shipment ID, exporter account, NFT serial, compliance data
EUDR_CERTIFICATE_TRANSFERRED	Ownership change	Shipment ID, from account, to account, NFT serial
EUDR_CERTIFICATE_FROZEN	Fraud detected	Account ID, NFT serial, reason
CUSTOMS_VERIFICATION	Customs check	Shipment ID, result, timestamp

Message Format

```
{
  "eventType": "EUDR_CERTIFICATE_ISSUED",
  "timestamp": "2025-10-25T10:30:00Z",
  "entityId": "SHP-2025-001",
  "entityType": "ComplianceCertificate",
  "data": {
    "shipmentId": "SHP-2025-001",
    "exporterAccountId": "0.0.123456",
    "nftSerialNumber": 1,
    "certificateType": "EUDR_COMPLIANCE",
    "issuedAt": "2025-10-25T10:30:00Z",
    "complianceData": {
      "originCountry": "Kenya",
      "riskLevel": "LOW",
      "totalFarmers": 45,
      "gpsCoordinatesCount": 67,
      "deforestationStatus": "VERIFIED_FREE"
    }
  }
}
```


Error Handling & Edge Cases

Scenario 1: Hedera Network Down During Registration

Problem: Cannot create Hedera account during entity registration

Solution: Graceful degradation

```
val hederaAccount = try {
    hederaAccountService.createHederaAccount(...)
} catch (e: Exception) {
    println("Failed to create Hedera account: ${e.message}")
    null // Entity still created without blockchain ID
}
```

Result:

- Entity created successfully
- hedera_account_id = null
- Can be populated later via background job

Scenario 2: Certificate Transfer Fails

Problem: Network error during NFT transfer

Solution: Retry mechanism

```
fun transferWithRetry(
    fromAccount: AccountId,
    toAccount: AccountId,
    shipmentId: String,
    maxRetries: Int = 3
): Boolean {
    repeat(maxRetries) { attempt ->
        try {
            return transferComplianceCertificateNft(fromAccount, toAccount, shipmentId)
        } catch (e: Exception) {
            if (attempt == maxRetries - 1) throw e
            Thread.sleep(1000 * (attempt + 1)) // Exponential backoff
        }
    }
    return false
}
```

Scenario 3: Importer Doesn't Have Associated NFT

Problem: Certificate transfer fails because importer hasn't associated token

Solution: Auto-associate during account creation

```
// In ImporterService.createImporter()
val nftId = hederaTokenService.getEudrComplianceCertificateNftId()
if (nftId != null) {
    hederaAccountService.associateTokenWithAccount(
        accountId,
        encryptedPrivateKey,
        nftId
    )
}
```

Scenario 4: Certificate Frozen by Mistake

Problem: Admin freezes wrong certificate

Solution: Unfreeze capability with audit trail

```
fun unfreezeComplianceCertificateNft(
    accountId: AccountId,
    adminId: String,
    reason: String
): Boolean {
    // Require admin authorization
    val admin = getAdmin(adminId)
    if (!admin.hasPermission("UNFREEZE_CERTIFICATE")) {
        throw UnauthorizedException()
    }

    // Unfreeze
    val success = hederaTokenService.unfreezeComplianceCertificateNft(accountId)

    // Record on HCS
    if (success) {
        hederaConsensusService.recordCertificateUnfreeze(
            accountId,
            adminId,
            reason
        )
    }

    return success
}
```

Scenario 5: Multiple Certificates for Same Shipment

Problem: Duplicate certificate issuance attempted

Solution: Check before issuing

```

fun issueComplianceCertificateNft(...): String {
    // Check if certificate already exists
    val existingCert = shipmentRepository.findById(shipmentId)
    if (existingCert.complianceCertificateNftId != null) {
        throw IllegalStateException("Certificate already issued for shipment $shipmentId")
    }

    // Proceed with issuance...
}

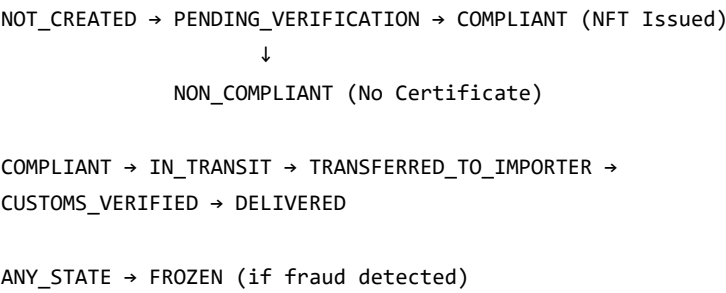
```

Summary

Key Takeaways

1. **Certificate Purpose:** Mandatory proof of EUDR compliance, NOT an incentive
2. **One Certificate Per Shipment:** Each compliant shipment gets ONE unique NFT
3. **Transferable:** Certificate follows physical goods ownership
4. **Immutable:** Blockchain provides permanent audit trail
5. **Revocable:** Can be frozen if fraud detected
6. **Verifiable:** Anyone can verify authenticity on public blockchain

State Flow Summary



Technical Architecture Summary

- **Account Creation:** Automatic during entity registration
- **Private Key Security:** AES-256-GCM encryption
- **NFT Association:** Automatic during account creation
- **Certificate Issuance:** When shipment passes all EUDR checks
- **Certificate Transfer:** When ownership changes
- **Blockchain Recording:** All events recorded on Hedera Consensus Service
- **Public Verification:** Anyone can verify on HashScan

Document Version: 1.0

Last Updated: October 25, 2025

Status: Production-Ready Architecture

Next Phase: Integration with shipment verification flow