SDG BLOCKCHAIN ACCELERATOR

Technical Architecture Document – zenGate Global Ltd

1. Project Information

- **Project Name:** zenGate Global - Traceability via Winter Protocol
- **Challenge & UNDP Office:** UNDP Bangladesh
- **Document Version:** v1.0

2. Overview

The **Winter Protocol API Service Pilot** addresses a critical adoption barrier in Bangladesh's traceability sector: existing platforms such as **SERA** are cloud-first and not blockchain-native, which limits trust, transparency, and long-term interoperability.

This project implements a **plug-and-play blockchain gateway** for traceability platforms, enabling them to **anchor verified events on Cardano** via APIs and JSON uploads, without requiring blockchain expertise. By abstracting wallet, key management, and transaction complexity, the service makes blockchain integration accessible to non-technical platform developers, UNDP staff, and other stakeholders.

3. System Architecture Diagram

What happens: *A traceability platform (e.g., SERA) calls the Winter API with a JSON event. The backend uploads the JSON to IPFS, hashes the CID into the on-chain datum, builds/submits the Cardano transaction, and returns a confirmation. Off-chain services index and surface results to dashboards inside the trace system or via a trace explorer.*

Components to include:

- **User Wallets:** *Abstracted behind Winter API* (service wallet signs; no direct key handling by SERA/UNDP users).
- **Aiken Validator (smart contract) Scripts:** Event lifecycle (create, recreate/update, spend/delete).
- **Off-chain components:** Typescript,Redis queue, Postgres DB, IPFS, Ogmios/Kupo indexers, dashboards.
- **Cardano Ledger (UTxO):** Anchors the hash of the IPFS CID in datum + event singleton.

```
 Traceability Platform (SERA)
           |
           | 1) POST /api/events { JSON }
           v
+--------------------+       2) Pin JSON      +---------+
|  Winter Backend    |----------------------->| IPFS    |
|  (API + Tx Builder)|<---CID-----------------+---------+
|   + Redis Queue    |
+--------------------+
           |
           | 3) Build Tx (datum: hash(CID), meta, version)
           v
+--------------------+   4) Validate/Submit   +---------------------+
| Aiken Validators   |----------------------->| Cardano Ledger UTxO |
| (event lifecycle)  |<--- 5) TxId/Conf ----->|  (NFT + datum)      |
+--------------------+                        +---------------------+
           |
           | 6) Monitor/Index (Ogmios/Kupo)
           v
+--------------------+        7) Persist refs/JSON copy (optional)
| Dashboards / APIs  |<-----------------------------------------------+
+--------------------+                     Database
```

4. Blockchain Design

Core Codebases

The traceability protocol is implemented across three open-sourced repositories:

1. **Winter-Cardano-Contracts**
   Aiken-based Plutus V2 smart contracts for Cardano. Provides the on-chain logic for creating, updating, and retiring stateful traceability events.
   Repo: https://github.com/zenGate-Global/winter-cardano-contracts

2. **Winter-Cardano (Library)**
   A TypeScript/JavaScript library that compiles and consumes the contracts. Exposes safe builders to instantiate events, pin metadata to IPFS, mint singleton tokens, recreate/update event state, and spend/retire events.
   Repo: https://github.com/zenGate-Global/winter-cardano/tree/main

3. **Winter-Backend (Cardano Service)**
   A backend service (PoC/MVP) that wraps the library into APIs. Handles IPFS pinning, transaction building, signing, submission, and indexing into a relational database. Includes retry logic, monitoring, and integration guides.
   Repo: https://github.com/zenGate-Global/winter-backend-cardano

Together, these repos separate **on-chain rules** (contracts), **transaction orchestration** (library), and **integration APIs** (backend service).

Smart Contracts (Winter-Cardano-Contracts)

Two main validators govern the lifecycle:

1. **Singleton Minting Policy (`singleton`)**
   - Purpose: Ensures only one unique NFT (the "singleton") can be minted per object.
   - Redeemers:
     - **Mint:** Requires consuming a predefined UTxO, guarantees uniqueness.
     - **Burn:** Required when the object is destroyed.
2. **Object Event Validator (`object_event`)**

    - Purpose: Manages the stateful UTxO holding the singleton token and on-chain datum.
    - Redeemers:
        - **RecreateEvent:** Updates the object by consuming the old UTxO and creating a new one with updated datum. A protocol fee is paid.
        - **SpendEvent:** Destroys the object, burning the singleton token. A protocol fee is paid.

Datum Structure (ObjectDatum)

```
pub type ObjectDatum {
  protocol_version: Int,
  data_reference: ByteArray,
  event_creation_info: Hash<Blake2b_256, Transaction>,
  signers: List<VerificationKeyHash>,
}
```

- **protocol_version** — Allows for upgrades.
- **data_reference** — Hash or reference to off-chain JSON metadata (e.g., IPFS CID).
- **event_creation_info** — TxId of the first create, permanent anchor of origin.
- **signers** — List of authorized transaction signers.

Redeemer Structure

- **Mint (singleton policy)** — Mint exactly one singleton. Done during creation of the event.
- **Burn (singleton policy)** — Burn when retiring. Part of the spend event.
- **RecreateEvent (object_event)** — Update `data_reference` while preserving other fields.
- **SpendEvent (object_event)** — Retire the object.

UTxO Model Usage

- **Create:** Mint singleton + lock stateful UTxO at object_event validator with datum and minimum ADA.
- **Recreate:** Consume old UTxO, output new UTxO at same address, singleton persists, `data_reference` changes, `event_creation_info` preserved.
- **Spend:** Consume UTxO, burn singleton, close contract.

Rules enforced by validator:

- At least one authorized signer must be the transaction creator.
- Singleton must remain with UTxO, unless its being spent
- Lovelace preserved or increased.
- Protocol fee paid on recreate and spend.

Token Management

- **Singleton Token (NFT):** Serves as the unique identifier key for the object.
- **Lifecycle:** Minted once, transferred during recreations, burned on spend.
- **Prevents double-counting:** Only one stateful UTxO can exist for each singleton.

Security Considerations

- **Authorized Signatures:** Transactions must be signed by keys listed in datum.
- **Replay Protection:** Singleton NFT uniqueness + `event_creation_info` field anchor.
- **Data Integrity:** Datum rules enforce that only `data_reference` changes on recreate.
- **Protocol Fees:** Mandatory payments disincentivize spam and support service sustainability.
- **Backend Hardening:** Winter-Backend enforces JSON schema validation, retries failed submissions, logs all attempts, and supports Postgres + Redis backups.

5. Data Flow & Transaction Lifecycle

1. **User/Platform Action:** Traceability platform (e.g., SERA) sends JSON event via API.
2. **Backend Process:**
    a. Uploads JSON to IPFS.
    b. Receives CID and hashes it.
    c. Constructs transaction with correct redeemer (create/recreate/spend).
3. **Validator Check:** Smart contracts validate datum structure, redeemer action, authorized signatures.
4. **Ledger Update:** UTxO created/updated/spent; singleton minted/moved/burned as required.

5. **Off-chain Monitoring:** Ogmios/Kupo index new transactions.
6. **Feedback:** Winter-Backend stores references in SQL and returns `TxId` + `CID` to platform/dashboard.

6. Off-chain Components

- **Winter-Backend:** API surface, transaction building, submission, retries, integration with IPFS and Cardano node.
- **Redis Queue:** Buffers requests, ensures reliable transaction processing.
- **Postgres Database:** Stores event metadata, transaction IDs, audit logs.
- **IPFS (e.g., Pinata, NFT.Storage):** Stores JSON payloads.
- **Cardano Node Integration:** Via Blockfrost, Kupmios, Maestro, or custom providers.
- **Ogmios / Kupo:** For transaction monitoring and indexing.
- **Dashboards/Reporting:** Demo UI for UNDP/stakeholders; APIs for traceability platform integration.

7. Sandbox/Testnet Results

- See explorer - https://zengate-traceit-dev.web.app/explorer
- Note - this is a beta version and not a public link (please do not share to community or public).

8. Tools and Environments Used

- **Cardano-node** (Preprod and Mainnet)
- **Aiken CLI** (`aiken build`, `aiken check`) for contracts
- **Winter-Cardano Library** for tx building
- **Winter-Backend** for API + orchestration
- **Ogmios** / **Kupo** for chain sync and monitoring

- **Redis** for queue management
- **Relational Database** (Postgres/MySQL) for indexing
- **IPFS** (Pinata / NFT.Storage) for JSON persistence
- **Docker + .env configs** for backend deployments

9. Remaining Considerations / Next Steps

- New Winter v2 Upgrades (incl. Multi-chain compatibility)
- IPFS Private Gateways
- Credential and data verification via Winter Auth - https://winter-auth.vercel.app/