

v3.4-EDG-002 – Decentralized Compute Graphs & Sovereign Twin Execution Chains

Document Title	Decentralized Compute Graphs & Sovereign Twin Execution Chains
Version	v3.4
Document ID	v3.4-EDG-002
Date	2025-03-22
Author	Take Back Your Data – Edge Architecture Group
Document Type	Public / Certification / Internal

1. Purpose & Scope

This document defines the use of decentralized compute graphs and sovereign twin execution chains in MaxOneOpen forks. It ensures that execution logic remains fully peer-driven, traceable, and certifiable across distributed edge environments.

2. Compute Graph Architecture

- Execution units (twins) form DAG-based compute graphs without central scheduler
- Task distribution is based on local availability, schema affinity and trust anchors
- Graph nodes must validate inbound and outbound edge operations via ZK-proofs
- Failure handling includes fallback chaining and sealed rollback logic

3. Twin Execution Chains

Chain Element	Execution Function	ZK Validation Logic
Initiator Twin	Root node of execution scope	Signed schema + token
Compute Node	Perform task / store output	Input/output trace match
Verifier Twin	Validate + acknowledge chain	Recursive replay hash
Audit Capsule	Snapshot + freeze trace	Chain closure hash

4. Certification Hooks

- All forks must support decentralized compute graph logic and twin chaining
- Execution chains must be cryptographically traceable and policy-bound
- Failure recovery must be deterministic and certifiable

5. Certification Triggers

- Centralized orchestration or opaque chain logic disqualifies fork
- Missing execution trace validation or fallback ambiguity breaks compliance

6. Certification Relevance

MaxOneOpen-certified forks must operate autonomously via distributed compute graphs and verified twin execution chains. Peer-driven, certifiable execution is a baseline requirement for edge-level certification.