

MaxAudit – Fork Registry Integration

Version: 1.0

Issued by: SAC, Take Back Your Data (TBYD)

License: MaxOneOpen License v2.2 – Structurally Enforced

1. Purpose

This document outlines the integration of MaxAudit with the Fork Registry. The goal is to ensure that every audit output references a specific, immutable Fork ID, enabling traceability across all variations of MaxOne-based infrastructures.

2. Definition of a Fork

A Fork is a legally and technically certified structural variant of a MaxOne instance. Each Fork is registered with a unique cryptographic identity and profile schema. Audits must be aligned to the exact Fork schema in use to be considered valid.

3. Registry Anchoring

- Every audit protocol includes a signed Fork ID reference
- The Fork ID is hash-anchored and timestamped within the audit chain
- Forks must be pre-registered in the global Fork Registry
- Operators may define private forks but lose audit eligibility unless registered

4. Verification Behavior

- During the audit, the system profile is matched against the Fork definition
- Deviations outside the Fork definition automatically escalate the audit status
- Fork mismatch is treated as a structural deviation (Level 2)
- Fork spoofing triggers a red audit result (tamper class)

5. Registry Interaction Logic

- Dongle: reads Fork ID locally; does not require live registry access
- Verifier: pulls Fork definition from offline mirror or preloaded cache
- Fork profile hash is validated against signed registry snapshot
- No external connectivity required for any validation step

6. Lifecycle & Governance

- Forks must be reviewed annually to remain in active registry
- Each Fork is bound to a specific operator or jurisdictional body
- Retired Forks remain verifiable for archival audits
- The registry is public, append-only, and cryptographically secured