# Module 02 – Fork Model & Structural Reproducibility

Module ID: CORE-FORK-002

Version: 4.1 (Revised CTO Edition)

Layer: Core Architecture Layer

Status: RELEASE

Dependencies: Module 00, Module 01

## 0. Purpose & Enforcement Point

This module defines the enforceable logic for system forks, structural reproducibility, and capsule-bound divergence control. It ensures that any derived or parallelized system instance is cryptographically bound to its origin, manifest-declared, and policy-verified. No forked system may operate independently of its Fork Capsule chain and Rule Inheritance Map.

## 1. Fork Invocation Interface & Control Object

Forking is only permitted via the `FORK\_CREATE()` protocol, which accepts:  
- `origin\_fork\_id` (or `null` for root);  
- `fork\_manifest\_id` (Module 06);  
- `execution\_scope\_map` (Module 03);  
- `inheritance\_mode` (strict, override, detached).  
Returns: `Fork Capsule ID` or `Fork Violation Capsule`.  
The Fork Control Object (FCO) binds every fork to a reproducible lineage structure and registers it in the Fork Chain Ledger (FCL).

## 2. Structural Logic, Inheritance Map & Scope Anchoring

Each fork must:  
- inherit or override the role, trust and policy bindings from the origin;  
- define its own Manifest Capsule;  
- commit its `Fork Lineage Capsule (FLC)` to the ledger.  
Fork Scopes are evaluated against:  
- `Role Execution Map` (Module 03);  
- `Governance Anchor Registry` (Module 04);  
- `Trust Scope Control` (Module 12).

## 3. Runtime Control, Merge Handling & Violation Hooks

At runtime, all forked sessions are isolated and governed by:  
- Manifest scope validation (Module 06);  
- Identity re-binding logic for forked actors;  
- Forbidden merge attempts without capsule-verified compatibility.  
Violation types:  
- orphaned fork without FCO binding: BLOCKED  
- unauthorized merge to origin: BLOCKED  
- lineage tampering: BLOCKED + Violation Ledger Entry

## 4. Capsule Schema & Forensic Artifacts

Each fork event generates the following:  
- Fork Lineage Capsule (FLC): `{fork\_id, origin\_id, manifest\_id, trust\_score, timestamp}`  
- Inheritance Map Capsule (IMC): `{policy\_ids, scope\_map, override\_flags}`  
- Fork Execution Envelope (FEE): `{session\_id, fork\_hash, exec\_result, violation\_flag}`  
These artifacts ensure reproducibility, lineage traceability, and structural audit integrity.

## 5. Intermodular Bindings & Trigger Anchors

This module links to:  
- Module 01 (Execution Gate) to authorize forked calls  
- Module 03 (Role & Scope Control) to enforce fork limits  
- Module 04 (Rule Anchoring) to bind inherited governance  
- Module 13 (LedgerSync) to record fork lineage  
- Module 14 (Audit Capsule) for forensic fork validation

## CTO Validation Matrix

Module 02 (CTO Edition) guarantees the following verifiable conditions:  
- All forks are bound to manifest and rule structures: YES  
- Forks cannot escape or override trust boundaries without capsule logic: YES  
- Each fork produces an immutable lineage capsule chain: YES  
- Unauthorized forks or merges are detected, blocked, and audited: YES  
- All fork executions are reproducible, traceable, and session-bound: YES