MaxTreaty v1.0 – Public Discovery, Registration and Fork Flagging

This document outlines the mechanisms used by MaxTreaty to publish, register, and track treaties over time. It also defines how forks and unauthorized duplications are flagged for inspection or rejection.

# 1. Treaty Discovery Protocol

MaxTreaty supports structured public discovery via:

* - Treaty registry APIs (REST, JSON-based)
* - Signed publication capsules (e.g., GitHub, IPFS, registry chains)
* - Metadata beacons embedded in capsules via `treaty.ref` and `treaty.anchor.ref`

# 2. Treaty Registration and Indexing

Public treaties may be indexed and versioned using:

* - Treaty registration manifest (`treaty.register.yaml`)
* - Embedded hash and version ID
* - Public ledger or MaxBridge-backed timeline references

# 3. Version Tracking

Treaties are versioned using hash-anchored chains with clear forward and backward links. Each treaty includes:  
- version.current  
- version.previous  
- optional version.aliases for public readability

# 4. Fork Detection

MaxTreaty flags forks using capsule lineage comparison. A treaty fork is declared if:  
- two treaties share a parent but diverge in class or scope  
- hash linkage is broken without valid anchor.ref  
- conflicting signers claim treaty succession without delegation

# 5. Fork Flagging and Response

When a fork is flagged:

* - the affected treaty is marked `treaty.status: forked`
* - MaxDeploy and MaxBridge may block its reference usage
* - MaxAudit tags the fork event in audit trail logs

# 6. Public Treaty Lifecycle Portal (optional)

MaxTreaty optionally supports treaty lifecycle publishing through a portal layer. Treaties can be browsed, verified, revoked or compared across forks using public UI or API endpoints.