MaxTreaty v1.0 – Signature Chains and Delegation Trees

This document defines the cryptographic signature model used in MaxTreaty and explains how authority can be delegated across roles and organizational boundaries. It includes rules for signature verification, delegation chain validation, and key revocation.

# 1. Signature Principles

All treaties must be cryptographically signed by their issuer using a registered MaxOneOpen-compatible signing identity. Each signature includes:  
- issuer.publicKey  
- signature.timestamp  
- hash.lock of the treaty content  
- optional revocation.key reference

# 2. Delegation Trees

A treaty issuer can delegate authority to subordinate roles using structured delegation trees. These are defined in `delegation.tree.json` and may include:  
- signer → admin → contributor  
- authority.root → subproject.treaty → capsule.issuer  
- org.certifier → compliance.node → registry.notary

# 3. Validation Logic

The MaxTreaty validation engine ensures:

* - Signature hash matches treaty content
* - Signer exists in registered ledger or delegation chain
* - Timestamp and expiration match governance scope
* - No revocation flag on signer or upstream delegate

# 4. Cross-System Delegation

Delegations are valid across capsule systems (Bridge, Deploy, Audit) if verifiable via anchor.ref or public delegation proof bundle. This allows trusted operators from one system to participate in treaty signing or validation in another, under strict control.

# 5. Revocation Handling

If a key or signer is revoked, MaxTreaty can:  
- Invalidate associated treaties  
- Notify linked systems (via MaxBridge or MaxAudit)  
- Mark capsule or delegation chain as broken