1. Scope

Go to Top

This part defines a distributed, node-based Logical Architecture for the Financial Systems Archetype as a Platform Independent Model.

This part specialises the Conceptual Architecture defined in Part 1: Conceptual Architecture into platform-independent logical elements, relationships, responsibilities, constraints, information flows, interaction patterns, runtime-plane relationships, governance relationships, and traceability relationships. This part also defines the logical basis for policy-governed information sharing, sovereignty-aware release, residency-aware movement, metadata and aggregate view generation, release control, and release Evidence. It identifies how the Logical Policy and Release Plane supports distributed enforcement, replicated decision capability, policy administration, policy information sources, obligations, release-specific exchanges, and audit/evidence relationships without prescribing a specific policy engine, gateway product, communication technology, or deployment topology.

This part applies to financial structured-information processing systems that use identifiable logical Nodes to communicate through explicit logical Communication Endpoints. It defines the Logical Architecture before later parts select communication technologies, data serialisation formats, programming languages, deployment mechanisms, runtime infrastructure, test mechanisms, and evidence-capture mechanisms.

This part defines one Logical Architecture that conforms to Part 1. It does not define every possible Logical Architecture that conforms to the Financial Systems Archetype Conceptual Architecture.

This part does not define the FX Demo Logical Profile. Part 3: FX Demo Logical Profile defines that profile. This part does not define DDS mappings, IDL structures, QoS profiles, Python implementation patterns, Protocol Buffers mappings, REST bindings, RPC bindings, container-image mappings, Kubernetes or K3s deployment, Crucible scenarios, acceptance checks, scripts, repository structure, or runtime evidence. Later parts and supporting repository documentation address those concerns.

This part guides authors and reviewers in preserving logical meaning before later parts add domain-specific, implementation, deployment, testability, or evidence details to the architecture.