

Navigation

- [Home](#)

Public Area

- [Public Area Home](#)

Private Area

- [Private Area Home](#)

FX Demo

- [FX Demo Home](#)
- [Part 0: Document Set Overview and Reader's Guide](#)
 - [Foreword](#)
 - [Introduction](#)
 - [1. Scope](#)
 - [2. Purpose of the Document Set](#)
 - [3. Relationship to SDIS/SIP](#)
 - [4. Layered Document Structure](#)
 - [5. Reading the Architecture as an Onion](#)
 - [6. Document Responsibilities](#)
 - [7. Layering Rules](#)
 - [7.1 Concepts Precede Logical Models](#)
 - [7.2 Logical Models Precede Implementation Mappings](#)
 - [7.3 Implementation Profiles: Realize Logical Models](#)
 - [7.4 Deployment Profiles Execute Implementation Profiles](#)
 - [7.5 Evidence Supports Claims](#)
 - [7.6 Runtime Identifiers Do Not Replace Architectural Concepts](#)
 - [7.7 Later Parts Must Preserve Traceability](#)
 - [8. Navigation Guidance](#)
 - [9. Status of Existing FX Demo and Phase 0 Material](#)
 - [10. Document Set Requirements](#)
- [Part 1: Conceptual Architecture](#)
 - [Foreword](#)
 - [Introduction](#)
 - [1. Scope](#)
 - [2. Relationship to Part 0](#)
 - [3. Traceability to Parent and Source Architectures](#)
 - [3.1 Traceability to SIP-RA](#)
 - [3.2 Alignment with FDIS-RA](#)
 - [3.3 Coverage of the Original FX Demo Reference Architecture](#)

- 3.4 Traceability Position
- 4. Conceptual Architecture Principles
 - 4.1 Separation of Concerns
 - 4.2 Platform Independence
 - 4.3 Classification Before Implementation
 - 4.4 Logical Meaning Before Physical Realization
 - 4.5 Traceability
 - 4.6 Evidence Orientation
 - 4.7 Governance of Meaning
- 5. Archetype Classification Model
 - 5.1 Overview
 - 5.2 Ecosphere
 - 5.3 Ecosystem
 - 5.4 Domain
 - 5.5 Classification Path
 - 5.6 Classification Values and Runtime Identifiers
 - 5.7 Classification and Governance
- 6. Financial Systems Classification
 - 6.1 Overview
 - 6.2 Finance Ecosphere
 - 6.3 Monetary Ecosystem
 - 6.4 Foreign Exchange Domain
 - 6.5 Classification Inheritance for FX Profiles
 - 6.6 Future Financial Classifications
 - 6.7 Classification Governance
- 7. Core Conceptual Elements
 - 7.1 Overview
 - 7.2 Node
 - 7.3 Node Identity
 - 7.4 Node Role
 - 7.5 Runtime Plane
 - 7.6 Communication Endpoint
 - 7.7 Data Structure Definition
 - 7.8 Data Structure Instance
 - 7.9 Evidence
 - 7.10
- 8. Conceptual Runtime Plane Model
 - 8.1 Overview
 - 8.2 Control Plane
 - 8.3 Data Plane
 - 8.4 Health and Observability Plane
 - 8.5 Policy And Release Plane
 - 8.6 Audit And Provenance Plane
 - 8.7 Cross-Plane Relationships
- 9. Separation of Concerns Model
 - 9.1 Overview
 - 9.2 Data
 - 9.3 Structure
 - 9.4 Semantics
 - 9.5 Interpretation
 - 9.6 Governance and Authority

- 9.7 Traceability and Evidence
 - 9.8 Cross-Concern Relationships
- 10. Conceptual Traceability Model
 - 10.1 Overview
 - 10.2 Traceability Across Layers
 - 10.3 Conceptual Element to Logical Element / PIM
 - 10.4 Logical Element / PIM to Implementation Artifact / PSM
 - 10.5 Implementation Artifact / PSM to Deployment Artifact
 - 10.6 Deployment Artifact to Evidence
 - 10.7 Bidirectional Review and Impact Analysis
 - 10.8 Traceability Boundaries
- 11. Conceptual Rules and Constraints
 - 11.1 Overview
 - 11.2 Conceptual Elements Remain Distinct
 - 11.3 Conceptual Architecture Does Not Prescribe Realization Mechanisms
 - 11.4 Classification Values Do Not Define Runtime Boundaries
 - 11.5 Runtime Planes Remain Conceptual Classifications
 - 11.6 Implementation Mappings Do Not Redefine Concepts
 - 11.7 Deployment Artifacts Do Not Redefine Implementation Artifacts
 - 11.8 Evidence Supports Claims
 - 11.9 Traceability Preserves Layering
 - 11.10 Local Profiles Must Preserve Parent Meaning
 - 11.11 Conceptual Constraints Summary
- 12. Relationship to the Logical Architecture / PIM
 - 12.1 Overview
 - 12.2 Conceptual Elements Become Logical Elements
 - 12.3 Logical Architecture Preserves Conceptual Meaning
 - 12.4 Logical Architecture Remains Platform Independent
 - 12.5 Logical Architecture Supports Domain Profiles
 - 12.6 Logical Architecture Provides the Basis for Implementation Profiles
 - 12.7 Traceability from Part 1 to Part 2
 - 12.8 Boundary Between Part 1 and Part 2
- 13. Conceptual Requirements
 - 13.1 Overview
 - 13.2 Conceptual Separation Requirements
 - 13.3 Classification Requirements
 - 13.4 Core Conceptual Element Requirements
 - 13.5 Runtime Plane Requirements
 - 13.6 Separation of Concerns Requirements
 - 13.7 Traceability Requirements
 - 13.8 Evidence Requirements
 - 13.9 Relationship to Logical Architecture / PIM Requirements
 - 13.10 Profile and Layering Requirements
- Part 2: Distributed Node-Based Logical Architecture
 - Foreword
 - Introduction
 - 1. Scope
 - 2. Relationship to Part 0 and Part 1
 - 3. Traceability to Parent and Source Architectures
 - 3.1 Traceability to SIP-RA
 - 3.2 Alignment with FDIS-RA

- 3.3 Traceability to Part 1 Conceptual Architecture
- 3.4 Coverage of the Original FX Demo Reference Architecture
- 3.5 Traceability Position
- 4. Logical Architecture Principles
 - 4.1 Distributed Node-Based Architecture
 - 4.2 Explicit Communication over Implicit Coupling
 - 4.3 Platform Independence
 - 4.4 Logical Separation of Runtime Planes
 - 4.5 Explicit Information Structures
 - 4.6 Traceability from Concept to Logical Model
 - 4.7 Evidence-Oriented Logical Design
 - 4.8 Policy-Governed Sharing Without Centralised Data Routing
- 5. Logical Architecture Overview
 - 5.1 Overview
 - 5.2 Logical Architecture Context
 - 5.3 Distributed Logical Node Network
 - 5.4 Logical Communication Model
 - 5.5 Logical Information Model
 - 5.6 Logical Runtime Plane Model
 - 5.7 Logical Traceability Model
- 6. Logical Node Model
 - 6.1 Overview
 - 6.2 Logical Node
 - 6.3 Logical Node Identity
 - 6.4 Logical Node Role
 - 6.5 Logical Node Responsibility
 - 6.6 Logical Node Boundary
 - 6.7 Logical Node Collaboration
 - 6.8 Logical Node Lifecycle State
- 7. Logical Communication Model
 - 7.1 Overview
 - 7.2 Logical Communication Endpoint
 - 7.3 Communication Role
 - 7.4 Publisher Role
 - 7.5 Subscriber Role
 - 7.6 Requester Role
 - 7.7 Responder Role
 - 7.8 Command Producer Role
 - 7.9 Command Consumer Role
 - 7.10 Event Producer Role
 - 7.11 Event Consumer Role
 - 7.12 Communication Direction and Coupling
 - 7.13 Communication Pattern Independence
- 8. Logical Information Model
 - 8.1 Overview
 - 8.2 Logical Data Structure Definition
 - 8.3 Logical Data Structure Instance
 - 8.4 Logical Message
 - 8.5 Logical Event
 - 8.6 Logical Command
 - 8.7 Logical Acknowledgement

- 8.8 Logical Assertion
- 8.9 Logical Record
- 8.10 Logical Information Lineage
- 9. Logical Runtime Plane Model
 - 9.1 Overview
 - 9.2 Logical Control Plane
 - 9.3 Logical Data Plane
 - 9.4 Logical Health and Observability Plane
 - 9.5 Logical Policy and Release Plane
 - 9.6 Logical Audit and Provenance Plane
 - 9.7 Cross-Plane Logical Relationships
- 10. Logical Interaction Patterns
 - 10.1 Overview
 - 10.2 Register Node
 - 10.3 Report Node Status
 - 10.4 Publish Domain Information
 - 10.5 Subscribe to Domain Information
 - 10.6 Issue Control Command
 - 10.7 Acknowledge Control Command
 - 10.8 Request Policy Decision
 - 10.9 Enforce Policy Decision and Produce Authorized Release
 - 10.10 Record Audit and Provenance
 - 10.11 Replay or Reconstruct Information
 - 10.12 Logical Interaction Pattern Summary
- 11. Logical Governance Model
 - 11.1 Overview
 - 11.2 Logical Ownership of Definitions
 - 11.3 Logical Versioning
 - 11.4 Logical Compatibility
 - 11.5 Logical Change Control
 - 11.6 Logical Admission Rules
 - Logical Policy Constraints
- 12. Logical Traceability Model
 - 12.1 Overview
 - 12.2 Part 1 Concept to Part 2 Logical Element
 - 12.3 Logical Node Traceability
 - 12.4 Logical Endpoint Traceability
 - 12.5 Logical Data Structure Traceability
 - 12.6 Logical Runtime Plane Traceability
 - 12.7 Logical Interaction Traceability
 - 12.8 Logical Evidence Traceability
 - 12.9 Logical Traceability Summary
- 13. Relationship to Domain Logical Profiles
 - 13.1 Overview
 - 13.2 How Domain Profiles Specialize the Logical Architecture
 - 13.3 Domain Profile Additions
 - 13.4 What Domain Profiles Preserve
 - 13.5 Relationship to the FX Demo Logical Profile
 - 13.6 Boundary Between Part 2 and Domain Logical Profiles
- 14. Relationship to Implementation Profiles / PSM
 - 14.1 Overview

- 14.2 Logical Elements Prepared for Technology Mapping
 - 14.3 Communication Technology Independence
 - 14.4 Data Representation Independence
 - 14.5 Execution and Deployment Independence
 - 14.6 Implementation Mapping Boundaries
 - 14.7 Relationship to Phase 0 Implementation Profile
 - 14.8 Boundary Between Part 2 and Implementation Profiles
- 15. Logical Rules and Constraints
 - 15.1 Overview
 - 15.2 Logical Nodes Remain Distinct from Runtime Deployments
 - 15.3 Logical Endpoints Remain Distinct from Communication Technologies
 - 15.4 Logical Data Structures Remain Distinct from Serialisation Formats
 - 15.5 Logical Runtime Planes Remain Distinct from Implementation Groupings
 - 15.6 Logical Interactions Remain Traceable
 - 15.7 Logical Architecture Does Not Prescribe Deployment Topology
 - 15.8 Logical Architecture Does Not Prescribe Communication Pattern Implementation
 - 15.9 Logical Architecture Does Not Prescribe Governance Tooling
 - 15.10 Domain Profiles Preserve the Logical Architecture
 - 15.11 Implementation Profiles Preserve the Logical Architecture
 - 15.12 Logical Constraints Summary
- 16. Logical Requirements
 - 16.1 Overview
 - 16.2 Distributed Node-Based Architecture Requirements
 - 16.3 Logical Node Requirements
 - 16.4 Logical Communication Requirements
 - 16.5 Logical Information Requirements
 - 16.6 Logical Runtime Plane Requirements
 - 16.7 Logical Interaction Requirements
 - 16.8 Logical Governance Requirements
 - 16.9 Logical Traceability Requirements
 - 16.10 Profile and Implementation Boundary Requirements
- Part 3: FX Demo Logical Profile
 - Foreword
 - Introduction
 - 1. Scope
 - 2. Relationship to Parts 0, 1, and 2
 - 3. Traceability to Parent and Source Architectures
 - 3.1 Traceability to SIP-RA
 - 3.2 Alignment with FDIS-RA
 - 3.3 Traceability to Part 1 Conceptual Architecture
 - 3.4 Traceability to Part 2 Distributed Node-Based Logical Architecture / PIM
 - 3.5 Coverage of the Original FX Demo Reference Architecture
 - 3.6 Traceability Position
 - 4. FX Demo Logical Profile Principles
 - 4.1 Domain Specialization without Conceptual Redefinition
 - 4.1 Domain Specialisation without Conceptual Redefinition
 - 4.2 FX Domain Scope before Implementation
 - 4.3 Distributed FX Node Network
 - 4.4 Explicit FX Communication Endpoints
 - 4.5 FX Information Structures before Serialisation

- 4.6 Runtime Plane Separation in the FX Demo
- 4.7 Traceability from Concept to Logical Profile
- 4.8 Evidence-Oriented FX Logical Design
- 4.9 Phase Extension without FX Logical Redefinition
- 5. FX Demo Logical Profile Overview
 - 5.1 Overview
 - 5.2 FX Demo Classification Path
 - 5.3 FX Demo Logical Architecture Context
 - 5.4 FX Demo Logical Node Network
 - 5.5 FX Demo Logical Communication Model
 - 5.6 FX Demo Logical Information Model
 - 5.7 FX Demo Logical Runtime Plane Model
 - 5.8 FX Demo Logical Traceability Model
- 6. FX Demo Logical Node Model
 - 6.1 Overview
 - 6.2 FX Transaction Intake Node
 - 6.3 FX Validation Node
 - 6.4 FX Semantic Interpretation Node
 - 6.5 FX Contract State Node
 - 6.6 FX Cash-Flow Computation Node
 - 6.7 FX Policy and Release Node
 - 6.8 FX Audit and Provenance Node
 - 6.9 FX Health and Observability Node
 - 6.10 FX Registry and Coordination Node
 - 6.11 FX Replay and Reconstruction Node
 - 6.12 External Oversight Participant
- 7. FX Demo Logical Node Roles
 - 7.1 Overview
 - 7.2 Transaction Intake Role
 - 7.3 Structural Validation Role
 - 7.4 Semantic Interpretation Role
 - 7.5 Contract State Management Role
 - 7.6 Cash-Flow Computation Role
 - 7.7 Policy Evaluation Role
 - 7.8 Release Control Node
 - 7.9 Audit Recording Role
 - 7.10 Provenance Recording Role
 - 7.11 Health Reporting Role
 - 7.12 Registry and Coordination Role
 - 7.13 Replay and Reconstruction Role
 - 7.14 Oversight Consumption Role
- 8. FX Demo Logical Communication Model
 - 8.1 Overview
 - 8.2 FX Transaction Intake Endpoint
 - 8.3 FX Validation Result Endpoint
 - 8.4 FX Semantic Interpretation Endpoint
 - 8.5 FX Contract State Endpoint
 - 8.6 FX Cash-Flow Obligation Endpoint
 - 8.7 FX Policy Decision Endpoint
 - 8.8 FX Release Package Endpoint
 - 8.9 FX Audit Record Endpoint

- 8.10 FX Provenance Record Endpoint
- 8.11 FX Node Status Endpoint
- 8.12 FX Control Command Endpoint
- 8.13 FX FX Replay Request Endpoint
- 8.14 FX Replay Result Endpoint
- 9. FX Demo Logical Information Model
 - 9.1 Overview
 - 9.2 FX Transaction Candidate
 - 9.3 FX Transaction Record
 - 9.4 FX Validation Result
 - 9.5 FX Semantic Assertion
 - 9.6 FX Contract State
 - 9.7 FX Cash-Flow Obligation
 - 9.8 FX Policy Decision
 - 9.9 FX Release Package
 - 9.10 FX Audit Record
 - 9.11 FX Provenance Record
 - 9.12 FX Node Status
 - 9.13 FX Control Command
 - 9.14 FX Command Acknowledgement
 - 9.15 FX Replay Request
 - 9.16 FX Replay Result
- 10. FX Demo Runtime Plane Model
 - 10.1 Overview
 - 10.2 Logical Control Plane
 - 10.3 Logical Data Plane
 - 10.4 Logical Health and Observability Plane
 - 10.5 Logical Policy and Release Plane
 - 10.6 Logical Audit and Provenance Plane
 - 10.7 Cross-Plane Relationships in the FX Demo
- 11. FX Transaction and Contract Lifecycle Model
 - 11.1 Overview
 - 11.2 FX Transaction Lifecycle Status
 - 11.3 FX Contract State Transitions
 - 11.4 Validation-Controlled Transitions
 - 11.5 Interpretation-Controlled Transitions
 - 11.6 Cash-Flow Computation Triggers
 - 11.7 Policy and Release State Considerations
 - 11.8 Audit and Provenance State Records
- 12. FX Demo Logical Interaction Patterns
 - 12.1 Overview
 - 12.2 Register FX Logical Node
 - 12.3 Report FX Node Status
 - 12.4 Intake FX Transaction Candidate
 - 12.5 Validate FX Transaction Candidate
 - 12.6 Interpret FX Transaction Semantics
 - 12.7 Establish or Update FX Contract State
 - 12.8 Compute FX Cash-Flow Obligations
 - 12.9 Request FX Policy Decision
 - 12.10 Release FX Information to Oversight Participant
 - 12.11 Record FX Audit and Provenance

- 12.12 Replay or Reconstruct FX Information
 - 12.13 FX Logical Interaction Pattern Summary
- 13. FX Demo Logical Governance Model
 - 13.1 Overview
 - 13.2 FX Logical Ownership of Definitions
 - 13.3 FX Logical Versioning
 - 13.4 FX Logical Compatibility
 - 13.5 FX Logical Change Control
 - 13.6 FX Logical Admission Rules
 - 13.7 FX Logical Policy Constraints
 - 13.8 FX Logical Governance Summary
- 14. FX Demo Logical Traceability Model
 - 14.1 Overview
 - 14.2 Part 1 Concept to FX Logical Profile Element
 - 14.3 Part 2 Logical Element to FX Logical Profile Element
 - 14.4 FX Node Traceability
 - 14.5 FX Endpoint Traceability
 - 14.6 FX Information Structure Traceability
 - 14.7 FX Runtime Plane Traceability
 - 14.8 FX Interaction Traceability
 - 14.9 FX Evidence Traceability
 - 14.10 FX Logical Traceability Summary
- 15. Relationship to Implementation Profiles / PSM
 - 15.1 Overview
 - 15.2 FX Logical Elements Prepared for Technology Mapping
 - 15.3 FX Communication Technology Independence
 - 15.4 FX Data Representation Independence
 - 15.5 FX Execution and Deployment Independence
 - 15.6 Boundary Between Part 3 and Part 4
 - 15.7 Relationship to Phase 0 Implementation Profile
- 16. FX Demo Logical Rules and Constraints
 - 16.1 Overview
 - 16.2 FX Logical Nodes Remain Distinct from Runtime Deployments
 - 16.3 FX Logical Endpoints Remain Distinct from Communication Technologies
 - 16.4 FX Logical Data Structures Remain Distinct from Serialisation Formats
 - 16.5 FX Runtime Planes Remain Distinct from Implementation Groupings
 - 16.6 FX Logical Interactions Remain Traceable
 - 16.7 FX Demo Logical Profile Does Not Prescribe Deployment Topology
 - 16.8 FX Demo Logical Profile Does Not Prescribe Communication Pattern Implementation
 - 16.9 FX Demo Logical Profile Does Not Prescribe Governance Tooling
 - 16.10 FX Demo Logical Constraints Summary
- 17. FX Demo Logical Requirements
 - 17.1 Overview
 - 17.3 FX Logical Node Requirements
 - 17.3 FX Logical Node Requirements
 - 17.4 FX Logical Node Role Requirements
 - 17.5 FX Logical Communication Requirements
 - 17.6 FX Logical Information Requirements
 - 17.7 FX Runtime Plane Requirements
 - 17.8 FX Lifecycle Requirements

- 17.9 FX Logical Interaction Requirements
- 17.10 FX Logical Governance Requirements
- 17.11 FX Logical Traceability Requirements
- 17.12 Implementation Boundary Requirements
- Part 4: Phase 0 Implementation Profile / PSM
 - Foreword
 - Introduction
 - 1. Scope
 - 2. Relationship to Parts 0, 1, 2, and 3
 - 3. Traceability to Parent and Source Architectures
 - 3.1 Traceability to SIP-RA
 - 3.2 Alignment with FDIS-RA
 - 3.3 Traceability to Part 1 Conceptual Architecture
 - 3.4 Traceability to Part 2 Distributed Node-Based Logical Architecture / Platform-Independent Model (PIM)
 - 3.5 Traceability to Part 3 FX Demo Logical Profile
 - 3.6 Traceability to Phase 0 Implementation Source Material
 - 3.7 Traceability to the Phase 0 Developer Handbook
 - 3.8 Traceability Position
 - 4. Phase 0 Implementation Profile Overview
 - 4.1 Overview
 - 4.2 Phase 0 Implementation Scope
 - 4.3 Phase 0 Implementation Baseline
 - 4.4 Phase-Specific Nature of the Profile
 - 4.5 Relationship to the FX Demo Logical Profile
 - 4.6 Relationship to Later Development Phases
 - 4.7 Phase 0 Implementation Profile Summary
 - 5. Phase 0 Implementation Principles
 - 5.1 Overview
 - 5.2 Implementation Artifacts Realize Logical Elements
 - 5.3 Traceability Before Code
 - 5.4 Developer Handbook Before Coding
 - 5.5 Generated Artifacts Before Manual Adaptation
 - 5.6 Explicit Interfaces Before Hidden Coupling
 - 5.7 Runtime Plane Discipline in Implementation
 - 5.8 Observable Behavior Before Operational Evidence
 - 5.9 Minimal Phase 0 Scope Before Expansion
 - 5.10 Phase 0 Implementation Principles Summary
 - 6. Phase 0 Implementation Preconditions
 - 6.1 Overview
 - 6.2 Part 3 Working Baseline
 - 6.3 Phase 0 Developer Handbook
 - 6.4 Repository Baseline
 - 6.5 IDL and Generation Baseline
 - 6.6 Logging and Exception-Handling Baseline
 - 6.7 Traceability Baseline
 - 6.8 No-Code-Before-Handbook Constraint
 - 6.9 Phase 0 Implementation Preconditions Summary
 - 7. Phase 0 Technology Selection and Implementation Context
 - 7.1 Overview
 - 7.2 DDS as Phase 0 Communication Technology

- 7.3 IDL as Phase 0 Data Definition Mechanism
- 7.4 QoS Profiles as Phase 0 Communication Behavior Controls
- 7.5 Generated Types as Phase 0 Implementation Data Artifacts
- 7.6 Python as Phase 0 Implementation Language
- 7.7 Scripts as Phase 0 Build and Execution Helpers
- 7.8 Repository Structure as Phase 0 Implementation Organization
- 7.9 Technology Selection Boundaries
- 7.10 Phase 0 Technology Selection Summary
- 8. Mapping from FX Logical Nodes to Implementation Artifacts
 - 8.1 Overview
 - 8.2 Node Mapping Principles
 - 8.3 Implementation Artifact Categories for Nodes
 - 8.4 FX Transaction Intake Node Mapping
 - 8.5 FX Validation Node Mapping
 - 8.6 FX Semantic Interpretation Node Mapping
 - 8.7 FX Contract State Node Mapping
 - 8.8 FX Cash-Flow Computation Node Mapping
 - 8.9 FX Policy and Release Node Mapping
 - 8.10 FX Audit and Provenance Node Mapping
 - 8.11 FX Health and Observability Node Mapping
 - 8.12 FX Registry and Coordination Node Mapping
 - 8.13 FX Replay and Reconstruction Node Mapping
 - 8.14 External Oversight Participant Mapping
 - 8.15 FX Logical Node Mapping Summary
- 9. Mapping from FX Logical Communication Endpoints to DDS Topics
 - 9.1 Overview
 - 9.2 Topic Mapping Principles
 - 9.3 DDS Topic Naming Principles
 - 9.4 FX Transaction Intake Topic Mapping
 - 9.5 FX Validation Result Topic Mapping
 - 9.6 FX Semantic Interpretation Topic Mapping
 - 9.7 FX Contract State Topic Mapping
 - 9.8 FX Cash-Flow Obligation Topic Mapping
 - 9.9 FX Policy Decision Topic Mapping
 - 9.10 FX Release Package Topic Mapping
 - 9.11 FX Audit Record Topic Mapping
 - 9.12 FX Provenance Record Topic Mapping
 - 9.13 FX Node Status Topic Mapping
 - 9.14 FX Control Command Topic Mapping
 - 9.15 FX Replay Request Topic Mapping
 - 9.16 FX Replay Result Topic Mapping
 - 9.17 Endpoint-to-Topic Mapping Summary
- 10. Mapping from FX Logical Information Structures to IDL and Generated Types
 - 10.1 Overview
 - 10.2 Information Structure Mapping Principles
 - 10.3 IDL Structure Naming Principles
 - 10.4 Generated Type Naming Principles
 - 10.5 FX Transaction Candidate Mapping
 - 10.6 FX Transaction Record Mapping
 - 10.7 FX Validation Result Mapping
 - 10.8 FX Semantic Assertion Mapping

- 10.9 FX Contract State Mapping
- 10.10 FX Cash-Flow Obligation Mapping
- 10.11 FX Policy Decision Mapping
- 10.12 FX Release Package Mapping
- 10.13 FX Audit Record Mapping
- 10.14 FX Provenance Record Mapping
- 10.15 FX Node Status Mapping
- 10.16 FX Control Command Mapping
- 10.17 FX Command Acknowledgement Mapping
- 10.18 FX Replay Request Mapping
- 10.19 FX Replay Result Mapping
- 10.20 Information-Structure-to-IDL Mapping Summary
- 11. Phase 0 Runtime Plane Realization
 - 11.1 Overview
 - 11.2 Control Plane Realization
 - 11.3 Data Plane Realization
 - 11.4 Health and Observability Plane Realization
 - 11.5 Policy and Release Plane Realization
 - 11.6 Audit and Provenance Plane Realization
 - 11.7 Cross-Plane Implementation Relationships
 - 11.8 Runtime Plane Realization Summary
- 12. Phase 0 Interaction Pattern Realization
 - 12.1 Overview
 - 12.2 Interaction Realization Principles
 - 12.3 Register FX Logical Node
 - 12.4 Report FX Node Status
 - 12.5 Intake FX Transaction Candidate
 - 12.6 Validate FX Transaction Candidate
 - 12.7 Interpret FX Transaction Semantics
 - 12.8 Establish or Update FX Contract State
 - 12.9 Compute FX Cash-Flow Obligations
 - 12.10 Request FX Policy Decision
 - 12.11 Release FX Information to Oversight Participant
 - 12.12 Record FX Audit and Provenance
 - 12.13 Replay or Reconstruct FX Information
 - 12.14 Interaction Pattern Realization Summary
- 13. Phase 0 QoS Profile Model
 - 13.1 Overview
 - 13.2 QoS Profile Purpose
 - 13.3 QoS Profile Mapping Principles
 - 13.4 Control Plane QoS Profile
 - 13.5 Data Plane QoS Profile
 - 13.6 Health and Observability Plane QoS Profile
 - 13.7 Policy and Release Plane QoS Profile
 - 13.8 Audit and Provenance Plane QoS Profile
 - 13.9 QoS Profile Mapping Summary
 - 13.10 QoS Boundaries and Non-Goals
- 14. Phase 0 Repository and Build Artifact Model
 - 14.1 Overview
 - 14.2 Repository Organization Principles
 - 14.3 Source Directory Model

- 14.4 IDL Directory Model
- 14.5 Generated Code Directory Model
- 14.6 Configuration Directory Model
- 14.7 Script Directory Model
- 14.8 Documentation Directory Model
- 14.9 Traceability Artifact Model
- 14.10 Build Artifacts
- 14.11 Generated Artifacts
- 14.12 Repository Baseline and Non-Baseline Material
- 14.13 Repository and Build Artifact Summary
- 15. Phase 0 Developer Handbook Requirements
 - 15.1 Overview
 - 15.2 Purpose of the Developer Handbook
 - 15.3 Handbook Approval before Coding
 - 15.4 File-Header Requirements
 - 15.5 Generated-File Header Requirements
 - 15.6 Naming Convention Requirements
 - 15.7 Coding Style Requirements
 - 15.8 Logging Convention Requirements
 - 15.9 Exception-Handling Requirements
 - 15.10 Traceability Convention Requirements
 - 15.11 Repository Workflow Requirements
 - 15.12 Script Usage Requirements
 - 15.13 Review Checklist Requirements
 - 15.14 Developer Handbook Summary
- 16. Phase 0 Script and Execution Model
 - 16.1 Overview
 - 16.2 Script Role in Phase 0
 - 16.3 Prerequisite Checking
 - 16.4 Type Generation
 - 16.5 Build Preparation
 - 16.6 Node Execution
 - 16.7 Multi-Node Startup
 - 16.8 Multi-Node Shutdown
 - 16.9 Script Logging and Error Handling
 - 16.10 Script Approval and Baseline Status
 - 16.11 Script Model Summary
- 17. Phase 0 Implementation Governance
 - 17.1 Overview
 - 17.2 Implementation Ownership
 - 17.3 Implementation Versioning
 - 17.4 Implementation Compatibility
 - 17.5 Implementation Change Control
 - 17.6 Generated Artifact Governance
 - 17.7 Repository Governance
 - 17.8 Developer Handbook Governance
 - 17.9 Implementation Governance Summary
- 18. Phase 0 Implementation Traceability Model
 - 18.1 Overview
 - 18.2 Part 3 FX Logical Element to Part 4 Implementation Artifact
 - 18.3 Implementation Artifact to Part 3 FX Logical Element

- 18.4 Node Implementation Traceability
- 18.5 Topic Implementation Traceability
- 18.6 IDL and Generated Type Traceability
- 18.7 Runtime Plane Implementation Traceability
- 18.8 Script and Repository Traceability
- 18.9 Developer Handbook Traceability
- 18.10 Implementation Traceability Summary
- 19. Relationship to Deployment, Testability, and Evidence Plan
 - 19.1 Overview
 - 19.2 Boundary Between Part 4 and Part 5
 - 19.3 Implementation Artefacts Prepared for Deployment
 - 19.4 Implementation Artefacts Prepared for Testability
 - 19.5 Implementation Artefacts Prepared for Observation
 - 19.6 Implementation Artefacts Prepared for Evidence
 - 19.7 Part 5 Handoff
 - 19.8 Relationship to Deployment, Testability, and Evidence Plan Summary
- 20. Phase 0 Implementation Rules and Constraints
 - 20.1 Overview
 - 20.2 Implementation Artefacts Do Not Redefine Logical Elements
 - 20.3 DDS Topics Do Not Redefine Logical Communication Endpoints
 - 20.4 IDL Structures Do Not Redefine Logical Information Structures
 - 20.5 Generated Types Do Not Redefine Logical Information Structures
 - 20.6 Runtime Processes Do Not Redefine FX Logical Nodes
 - 20.7 Repository Layout Does Not Redefine Architecture
 - 20.8 Scripts Do Not Redefine Interaction Patterns
 - 20.9 Developer Handbook Does Not Replace the Implementation Profile
 - 20.10 Phase 0 Does Not Define Later Phase Implementation Profiles
 - 20.11 Deployment and Evidence Concerns Remain Outside Part 4
 - 20.12 Implementation Constraints Summary
- 21. Phase 0 Implementation Requirements
 - 21.1 Overview
 - 21.2 Phase 0 Profile Requirements
 - 21.3 Technology Selection Requirements
 - 21.4 Logical Node Mapping Requirements
 - 21.5 Communication Endpoint Mapping Requirements
 - 21.6 Information Structure Mapping Requirements
 - 21.7 Runtime Plane Realization Requirements
 - 21.8 Interaction Pattern Realization Requirements
 - 21.9 QoS Profile Requirements
 - 21.10 Repository and Build Artifact Requirements
 - 21.11 Developer Handbook Requirements
 - 21.12 Script and Execution Requirements
 - 21.13 Implementation Governance Requirements
 - 21.14 Implementation Traceability Requirements
 - 21.15 Part 5 Handoff Requirements
- Part 5: Phase 0 Developer Handbook
 - Foreword
 - Introduction
 - 1. Purpose and Scope
 - 1.1 Purpose of the Handbook
 - 1.2 Scope of Phase 0

- 1.3 Intended Audience
- 1.4 How to Use This Handbook
- 2. Relationship to the Architecture Documents
 - 2.1 Parent Architecture Sources
 - 2.2 Relationship to SIP-RA
 - 2.3 Relationship to FDIS-RA
 - 2.4 Relationship to Parts 1, 2, and 3
 - 2.5 Relationship to the Phase 0 Implementation Source Material
 - 2.6 Relationship to Generated Code and Runtime Artefacts
- 3. Phase 0 Baseline
 - 3.1 Purpose of the Phase 0 Baseline
 - 3.2 Logical Architecture Baseline
 - 3.3 FX Demo Logical Profile Baseline
 - 3.4 Implementation Technology Baseline
 - 3.5 Repository Baseline
 - 3.6 Baseline Change Control
- 4. Repository Layout
 - 4.1 Repository Design Principles
 - 4.2 Top-Level Repository Structure
 - 4.3 Source Directories
 - 4.4 Generated Artefact Directories
 - 4.5 Configuration Directories
 - 4.6 Script Directories
 - 4.7 Test Directories
 - 4.8 Documentation Directories
 - 4.9 Git-Controlled and Non-Git-Controlled Content
- 5. Development Environment
 - 5.1 Supported Operating Systems
 - 5.2 VS Code / Code
 - 5.3 Python Environment
 - 5.4 Java Environment
 - 5.5 DDS Tooling
 - 5.6 Docker / Container Environment
 - 5.7 Git Environment
 - 5.8 Environment Verification
- 6. Prerequisites and Tooling
 - 6.1 Required Tools
 - 6.2 Optional Tools
 - 6.3 Version Requirements
 - 6.4 Tool Installation Notes
 - 6.5 Prerequisite Validation
 - 6.6 Troubleshooting Tool Issues
 - 6.7 Tool Baseline Checklist
 - 6.8 Multiple Products, Platforms, and Implementations
 - 6.9 Tool Baseline Checklist Instances
- 7. Naming and File Conventions
 - 7.1 General Naming Principles
 - 7.2 Directory Naming
 - 7.3 Source File Naming
 - 7.4 Generated File Naming
 - 7.5 Script Naming

- 7.6 Configuration File Naming
- 7.7 Topic Naming
- 7.8 Node Naming
- 7.9 Container Naming
- 7.10 File Headers and Inline Documentation
- 7.11 Spelling, Grammar, and Style
- 8. Generated Artifacts
 - 8.1 Purpose of Generated Artifacts
 - 8.2 Generated Versus Handwritten Files
 - 8.3 IDL-Derived Artifacts
 - 8.4 Python-Generated Support Artifacts
 - 8.5 Generated Documentation
 - 8.6 Regeneration Rules
 - 8.7 Protecting Handwritten Code from Overwrite
- 9. DDS / IDL / Type Generation
 - 9.1 Role of IDL in Phase 0
 - 9.2 IDL Source Location
 - 9.3 Control Plane Type Definitions
 - 9.4 Data Plane Type Definitions
 - 9.5 Type Generation Workflow
 - 9.6 Generated Type Locations
 - 9.7 Type Generation Validation
 - 9.8 Type Compatibility Rules
- 10. Node Implementation Pattern
 - 10.1 Purpose of the Node Pattern
 - 10.2 Node Lifecycle
 - 10.3 Node Startup Behavior
 - 10.4 Node Runtime Behavior
 - 10.5 Node Shutdown Behavior
 - 10.6 Node Status Reporting
 - 10.7 Node Command Handling
 - 10.8 Node Configuration
 - 10.9 Node Error Handling
 - 10.10 Node Documentation Requirements
- 11. Control Plane Topics
 - 11.1 Purpose of the Control Plane
 - 11.2 Node Status Topic
 - 11.3 Node Command Topic
 - 11.4 Control Plane Message Semantics
 - 11.5 Control Plane QoS Expectations
 - 11.6 Control Plane Validation
 - 11.7 Control Plane Observability
- 12. Data Plane Topics
 - 12.1 Purpose of the Data Plane in Phase 0
 - 12.2 Placeholder Data Plane Topics
 - 12.3 FX Demo Data Topics
 - 12.4 Topic Participation Rules
 - 12.5 Data Plane QoS Expectations
 - 12.6 Data Plane Validation
 - 12.7 Data Plane Non-Goals for Phase 0
- 13. Build and Run Scripts

- 13.1 Purpose of Build and Run Scripts
- 13.2 Shell Script Conventions
- 13.3 Python Utility Script Conventions
- 13.4 createRepositoryStructure.sh
- 13.5 checkPrerequisites.sh
- 13.6 initialisePythonEnvironment.sh
- 13.7 generateTypes
- 13.8 buildContainers.sh
- 13.9 runAll.sh
- 13.10 runNode.sh
- 13.11 stopAll.sh
- 13.12 prepareGitRepository.sh
- 13.13 Script Exit Codes
- 13.14 Script Logging
- 14. Containerisation
 - 14.1 Purpose of Containerisation in Phase 0
 - 14.2 Container Design Principles
 - 14.3 Container Build Inputs
 - 14.4 Container Runtime Configuration
 - 14.5 Container Networking
 - 14.6 Container Naming
 - 14.7 Container Logs
 - 14.8 Container Cleanup
- 15. Logging and Observability
 - 15.1 Purpose of Logging
 - 15.2 Log Format
 - 15.3 Log Location
 - 15.4 Node Logs
 - 15.5 Script Logs
 - 15.6 Container Logs
 - 15.7 Control Plane Observation
 - 15.8 Evidence Collection
- 16. Exception Handling
 - 16.1 Purpose of Exception Handling Rules
 - 16.2 Node Exceptions
 - 16.3 Script Failures
 - 16.4 Configuration Errors
 - 16.5 DDS Communication Errors
 - 16.6 Container Errors
 - 16.7 Recoverable and Non-Recoverable Errors
 - 16.8 Error Reporting
- 17. Git and Review Workflow
 - 17.1 Repository Initialisation
 - 17.2 Branching Approach
 - 17.3 Commit Conventions
 - 17.4 Pull Request Expectations
 - 17.5 Review Checklist
 - 17.6 Generated File Review Rules
 - 17.7 Tagging the Phase 0 Baseline
 - 17.8 Review of Documentation and Comments
 - 17.9 Review Outcomes

- 17.10 Relationship to Issues and Decision Records
- 18. Acceptance Criteria
 - 18.1 Repository Acceptance Criteria
 - 18.2 Environment Acceptance Criteria
 - 18.3 Type Generation Acceptance Criteria
 - 18.4 Node Acceptance Criteria
 - 18.5 Control Plane Acceptance Criteria
 - 18.6 Data Plane Acceptance Criteria
 - 18.7 Container Acceptance Criteria
 - 18.8 Documentation Acceptance Criteria
 - 18.9 Evidence Acceptance Criteria
 - 18.10 Phase 0 Completion Criteria
- 19. Phase 0 Non-Goals
 - 19.1 Production Readiness
 - 19.2 Full ACTUS Processing
 - 19.3 Full Financial Semantics
 - 19.4 Full Security Model
 - 19.5 Full IEF Policy Enforcement
 - 19.6 Full Operational Dashboard
 - 19.7 Performance Optimization
 - 19.8 Enterprise Deployment
 - 19.9 Full Multi-Vendor Interoperability
 - 19.10 Complete Automation
- 20. Risks and Open Decisions
 - 20.1 Implementation Language Decisions
 - 20.2 DDS Vendor Decisions
 - 20.3 Python Scope Decisions
 - 20.4 Container Runtime Decisions
 - 20.5 Generated Artifact Governance
 - 20.6 Repository Ownership
 - 20.7 September 2026 Demo Dependencies
 - 20.8 Open Issues Register
 - 20.9 Decision Record Expectations
 - 20.10 Risk Review and Baseline Acceptance
- Part 6: Cost Recovery, Compensation, and Settlement Plane
 - Foreword
 - Introduction
 - 1. Scope
 - 2. Purpose
 - 3. Architectural Motivation
 - 4. Architectural Principle and Requirements
 - 5. Definitions
 - Cost Recovery, Compensation, and Settlement Plane
 - Chargeable Work Unit
 - Work Performed Event
 - Qualified Service Provider
 - Evidence Reference
 - Compensation Claim
 - Cost Rule
 - Settlement Instruction
 - Non-Performance Indicator

- Competitive Substitution
 - Content Minimization
 - Under-Compliance
 - 6. Relationship to Existing FX Pricing
 - 7. Relationship to Other Architectural Planes
 - 8. Information Minimization
 - 8.1 Information Minimization Requirements
 - 8.2 Minimum Cost-Relevant Information
 - 8.3 Protected Information
 - 9. Operating Model
 - 9.1 Core Concepts
 - 9.2 Chargeable Work Types
 - 9.3 Competition and Provider Substitution
 - 9.4 Accounting Basis
 - 9.5 Non-Performance Visibility
 - 10. FX Demo Application
 - 10.1 Use Case
 - 10.2 Conceptual Flow
 - 11. Platform-Independent Model View
 - 12. Platform-Specific Mapping
 - 13. Phase 0 and Later-Phase Treatment
 - 14. Minimum Demonstration Profile
 - 15. Governance Requirements
 - 16. Open Issues
 - 17. Summary
- Part 7: Governed Node Service Market
 - Foreword
 - Introduction
 - 1. Scope
 - 2. Purpose
 - 3. Architectural Motivation
 - 4. Relationship to Part 6
 - 5. Core Concept
 - 6. Stakeholders
 - Buyer
 - Integrator
 - Regulator
 - Auditor
 - Governance Body
 - 7. Non-Functional Characteristics
 - 7.1 Non-Functional Characteristic
 - 7.2 Portability
 - 7.3 Reliability
 - 7.4 Maintainability
 - 7.5 Securability
 - 7.6 Manageability
 - 7.7 Usability
 - 7.8 Performance
 - 7.9 Interoperability
 - 7.10 Elasticity
 - 7.11 Scalability

- 8. Qualification and Assurance Value
 - 8.1 Qualification Profile
 - 8.2 Qualification Evidence
- 9. Market Operating Model
 - 9.1 Market Participation
 - 9.2 Selection Criteria
 - 9.3 Qualification-Based Eligibility
 - 9.4 Competitive Substitution
 - 9.5 Evidence-Based Assignment and Review
- 10. Relationship to Cost Recovery, Compensation, and Settlement
 - 10.1 Service Compensation
 - 10.2 Compensation Eligibility
 - 10.3 Work Evidence and Compensation Claims
 - 10.4 Settlement Instructions
 - 10.5 Under-Compliance and Non-Performance Visibility
- 11. Market Integrity Controls
 - 11.1 Content Minimization
 - 11.2 Governance Rules
 - 11.3 Suspension and Removal
 - 11.4 Appeal and Review
- 12. Architectural Principles
 - 12.1 Qualified Participation
 - 12.2 Non-Functional Assurance
 - 12.3 Policy-Constrained Competition
 - 12.4 Competitive Substitution Without Bypass
 - 12.5 Evidence-Based Compensation
 - 12.6 Content Minimization
 - 12.7 Distributed Authority, Shared Discipline
- 13. Terms Introduced by Part 7
 - 13.1 Governed Node Service Market
 - 13.2 Qualification Profile
 - 13.3 Qualification Evidence
- 14. Summary

From:

<https://wiki.didosolutions.com/> - **Dido Solutions, Inc Wiki**

Permanent link:

<https://wiki.didosolutions.com/sidebar?rev=1782321302>

Last update: **2026/06/24 10:15**

