Module 6 Summary: The Infrastructure Problem

Prof. Dr. Joerg Osterrieder

Digital Finance — BSc Course

Module 6: Prerequisites

Theme: Infrastructure (payment rails, core banking, APIs, next-gen) **From prior modules:**

- **M1L1:** the payment-flow anatomy
- **M1L3:** real-time payment mechanics
- **M3L1:** cryptographic-foundations intuition for ledger design

External knowledge assumed:

- Networking primer: TCP, HTTP, REST verbs at the level of one paragraph
- Databases basics: relational vs. ledger, ACID, eventual consistency
- JSON parsing and a basic API call from Python

Will be introduced this module: We introduce the world's payment rails (ACH, RTGS, FedNow, ISO 20022), core-banking systems, the API economy (PSD2/PSD3, open banking), and next-generation infrastructure.

Prerequisites are advisory; lessons remain self-contained where feasible. Forward references inside lessons flag any concept used before its canonical introduction.

Key Concepts by Lesson

L1: Payment Rails

- ACH/SEPA: batch/DNS, hours–days, payroll and bill pay
- RTGS (Fedwire/TARGET2): real-time, large-value interbank
- SWIFT: messaging layer for cross-border instructions
- Gross (RTGS) vs. net (DNS) settlement trade-offs

L2: Core Banking Systems

- General ledger at the foundation; product engines above
- Legacy COBOL systems vs. cloud-native challengers
- Cloud migration trade-offs: latency, regulation, cost
- Global market: $18B (2024), growing 10%+ annually

L3: API Economy

- REST, GraphQL, webhooks: when to use each
- API gateway as control plane of modern banking
- Banking-as-a-Service (BaaS) and embedded finance
- Security layers: OAuth 2.0, mTLS, rate limiting

L4: Next-Gen Infrastructure

- CBDCs: retail vs. wholesale, account vs. token-based
- RWA tokenization: bonds, equities, real estate on-chain
- Design choices: centralized ledger vs. DLT
- Programmable money: expiry dates, spending restrictions

Module 6 answers: What pipes carry money, how old are they, and what will replace them?

Key Formulas and Definitions

Settlement Types

RTGS (Real-Time Gross Settlement): Each transaction settled individually in real time. Zero counterparty risk, high liquidity cost.
DNS (Deferred Net Settlement): Transactions netted and settled in batches. Lower liquidity need, but settlement risk during the netting window.

Netting Efficiency

$$\text{Net Position}_i = \sum_j \text{Payments Received}_{j \to i} - \sum_j \text{Payments Sent}_{i \to j}$$

Netting reduces total liquidity required. Example: if A owes B €100 and B owes A €80, only €20 settles.

Core Banking Architecture (5-Layer Model)

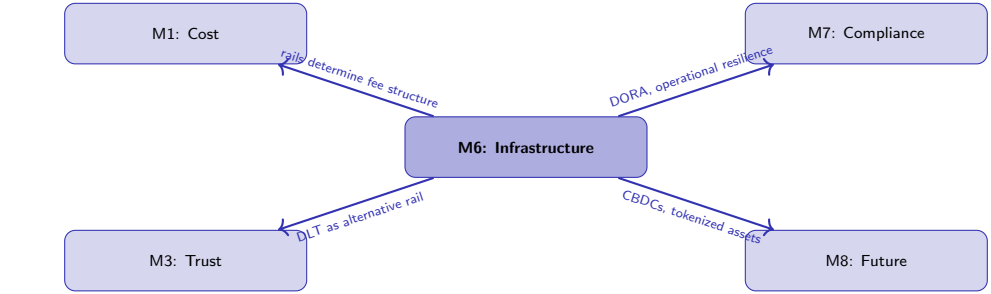
Channels (top) → API Gateway → Business Logic → Product Engines → General Ledger (bottom).
Every payment, balance, and interest calculation flows through this stack.

CBDC Design Taxonomy

Four design dimensions: (1) Retail vs. Wholesale, (2) Account-based vs. Token-based, (3) Centralized vs. Distributed ledger, (4) Intermediated vs. Direct access.

Infrastructure is invisible to customers but determines the speed, cost, and resilience of every financial transaction.

Connections to Other Modules



- **Infrastructure → Cost (M1):** Payment rail choice (RTGS vs. DNS vs. card network) determines the fee structure merchants and consumers face
- **Infrastructure → Trust (M3):** Distributed ledger technology is both a competitor to and a potential replacement for centralized payment rails
- **Infrastructure → Compliance (M7):** DORA mandates operational resilience for financial infrastructure; cloud outsourcing requires regulatory approval
- **Infrastructure → Future (M8):** CBDCs could replace the retail payment stack; RWA tokenization could replace settlement infrastructure

Today's infrastructure was built for yesterday's finance. The question is whether to renovate or rebuild.

Module 6: Integrative Questions

Two questions that need more than one lesson to answer:

1. M6L1 + M6L3 describe two cooperative-but-distinct payment infrastructures (wholesale rails + API economy). Name one private-data leakage failure that PSD3 (M6L3) creates that PSD2 did not.
2. Combine M6L2 (core banking) + M6L4 (next-gen infrastructure) to argue whether tokenised deposits are a strict upgrade or a strict alternative to traditional ledger banking.

Use these as study prompts before the module exam; each integrates concepts that span lessons.

Module 6: Worked Multi-Lesson Example

Worked example: route a cross-border tokenised-bond settlement through M6L1 (wholesale RTGS), M6L2 (core-banking GL booking), M6L3 (open-banking API consent flow for the buyer KYC), M6L4 (CBDC or SIX SDX delivery-versus-payment leg). **Pedagogical pattern:** the example is intentionally end-to-end. Solve it lesson-by-lesson, then step back and identify the lesson whose assumption was the binding constraint.

The exam-style version of this example appears in extttv4/exam_prep/exam_bank.tex for module 6.

Module 6: Forward-Pointing Dependency Map

Concepts from Module 6 that later modules will use:

- **M3L4:** Stablecoin rails (M3L4) extend M6L4 next-gen infrastructure with permissionless settlement
- **M7L3:** DORA (M7L3) is the operational-resilience framework that supervises everything M6 builds
- **M8L2:** Quantum risk (M8L2) threatens the cryptographic underpinnings of M6L1 + M6L4 in the same way

---

Forward-pointing dependencies; concepts not in this map are local to Module 6.