Digital Integration Hub: Architecture Playbook & Modernization Blueprint

A Practical Guide to Modern Digital Integration Architecture

Version 1.0

Executive Edition

A consulting-grade reference for enterprise architects, CTOs, and modernization leaders designing scalable, event-driven digital platforms.

Table of Contents

Typical Digital Integration Hub (DIH) Architecture Frameworks

A Digital Integration Hub is not a single product but an architectural pattern that combines *event-driven integration*, *API management*, and a *high-performance data layer* to decouple backend systems from digital channels.
The frameworks below represent the **most common architectural models** used by enterprises.

1. Canonical DIH Reference Architecture (Gartner Pattern)

(Most widely adopted conceptual model)

Core Components

- **API Gateway / API Management Layer**
  Exposes unified APIs to channels and partners.
- **High-Performance Data Store / In-Memory Grid**
  Often implemented with technologies like Apache Ignite, Redis, Hazelcast, or GigaSpaces.
  Provides low-latency access to aggregated data.
- **Event Streaming Backbone**
  Typically Kafka or Pulsar; handles CDC, domain events, and async updates.
- **Data Ingestion & Synchronization Layer**
  Connects to systems of record (SoR) via CDC, messaging, or scheduled sync.
- **Microservices Layer**
  Encapsulates business logic and orchestrates data flows.
- **Security & Governance Layer**
  Identity, access control, auditing, data lineage.

When used

Enterprises modernizing legacy systems without full replacement, needing scalable digital channels.

2. In-Memory Data Grid–Centric DIH (e.g., GigaSpaces, Apache Ignite)

(Optimized for ultra-low latency and real-time analytics)

Architectural Traits

- Distributed in-memory compute and storage
- Co-located processing for microservices
- Built-in CDC connectors
- Event-driven updates from SoR
- Often includes SQL, key-value, and document models

When used

High-throughput digital channels, financial services, telco, retail.

3. iPaaS-Driven DIH (Boomi, MuleSoft, SnapLogic)

(Integration-first approach with DIH capabilities layered on top)

Architectural Traits

- Centralized integration flows (ETL/ELT, API-led connectivity)
- Event-driven orchestration
- Optional data hub or data fabric layer
- Strong governance and lifecycle tooling

When used

Organizations with strong iPaaS adoption wanting DIH without building from scratch.

4. Data Fabric / Data Mesh–Aligned DIH (Cinchy, Dataware Platforms)

(Data-as-a-product meets DIH)

Architectural Traits

- Logical data layer decoupled from applications
- Metadata-driven governance
- Bi-directional sync with SoR
- Self-service data access for business teams
- Often includes semantic modeling

When used

Enterprises aiming for long-term data autonomy and cross-domain data sharing.

5. Microservices + Event Streaming DIH (Kafka-Centric)

(Lightweight DIH pattern built around event logs)

Architectural Traits

- Kafka as the system of record for events
- Materialized views stored in Redis, Elastic, or Postgres
- Microservices consume and update views
- API gateway exposes aggregated data

When used

Cloud-native organizations with strong DevOps maturity.

6. Platform-Specific DIH Frameworks (Vendor Architectures)

Vendors provide prescriptive DIH blueprints, e.g.:

- **GigaSpaces Smart DIH** – in-memory compute + microservices + CDC

- **Mia-Platform DIH** – API-first, microservices, event-driven, developer portal
- **Boomi Enterprise Platform DIH Pattern** – API + integration + event automation

These frameworks provide accelerators, governance, and prebuilt connectors.

🧩 Comparison Table: DIH Architecture Frameworks

Framework Type	Strengths	Weaknesses	Best For
Gartner Canonical DIH	Balanced, scalable, vendor-neutral	Requires custom assembly	Large enterprises modernizing legacy
In-Memory Grid DIH	Ultra-low latency, real-time	Higher cost, specialized ops	Financial services, telco, retail
iPaaS-Driven DIH	Fast to implement, strong governance	Less control over internals	Organizations with existing iPaaS
Data Fabric / Mesh DIH	Strong governance, data-as-product	Cultural shift required	Enterprise with data strategy focus
Kafka-Centric DIH	Cloud-native, scalable	Requires engineering maturity	Digital-native companies
Vendor DIH Platforms	Accelerated delivery, integrated stack	Vendor lock-in	Fast modernization projects

Canonical DIH Reference Architecture (Gartner Pattern)

1. High-level logical view

At its core, the DIH sits **between systems of record and digital channels**, providing a **decoupling, caching, and integration layer**:

- **Channels:** Web, mobile, partner APIs, internal apps
- **Experience/API Layer:** API gateway + API management
- **Digital Integration Hub Core:** High-performance data store + microservices + event streaming
- **Enterprise Integration Layer:** CDC, ESB/iPaaS, connectors

- **Systems of Record (SoR):** Core banking, ERP, CRM, mainframe, legacy apps

2. Layered architecture

2.1 Channel and experience layer

- **Purpose:** Serve digital channels with low-latency, stable APIs, independent of SoR constraints.
- **Key components:**
- **API Gateway / API Management:** Routing, throttling, versioning, developer portal.
- **Edge Security:** WAF, rate limiting, OAuth2/OIDC, JWT validation.

2.2 Digital services and orchestration layer

- **Purpose:** Encapsulate business capabilities and orchestrate data from the DIH data store and SoR.
- **Key components:**
- **Domain Microservices:** Customer, product, order, payments, etc.
- **Orchestration/Composition Services:** Combine multiple domain services into higher-level journeys.
- **Synchronous APIs:** REST/GraphQL exposed via the API gateway.
- **Policy Enforcement:** Authorization, data masking, consent checks.

2.3 DIH data hub layer (high-performance data store)

- **Purpose:** Provide **aggregated, denormalized, channel-optimized data** with low latency.
- **Key components:**
- **High-Performance Data Store:** In-memory grid, distributed cache, or fast NoSQL (e.g., Redis, Ignite, Hazelcast).

- **Materialized Views / Read Models:** Pre-computed views for key journeys (customer 360, product catalog, etc.).
- **Caching & TTL Policies:** Staleness rules, refresh strategies, cache invalidation.

2.4 Event streaming and data ingestion layer

- **Purpose:** Keep the DIH data hub in sync with SoR via **event-driven and CDC-based** mechanisms.
- **Key components:**
- **Event Streaming Platform:** Kafka/Pulsar for domain events and change events.
- **CDC Connectors:** Log-based CDC from RDBMS/mainframe to event streams.
- **Ingestion Services:** Transform, enrich, and load events into the DIH data store.
- **Replay & Recovery:** Rebuild materialized views from event logs.

2.5 Enterprise integration layer

- **Purpose:** Connect to SoR using appropriate integration styles.
- **Key components:**
- **ESB / iPaaS / Integration Platform:** For SOAP, file, batch, proprietary protocols.
- **Synchronous Adapters:** For real-time lookups when data is not in the hub.
- **Batch/ETL Jobs:** For non-real-time data domains.

2.6 Systems of record layer

- **Purpose:** Authoritative systems for data and transactions.
- **Examples:**
- Core banking, policy admin, ERP, CRM, billing, mainframe apps, line-of-business systems.

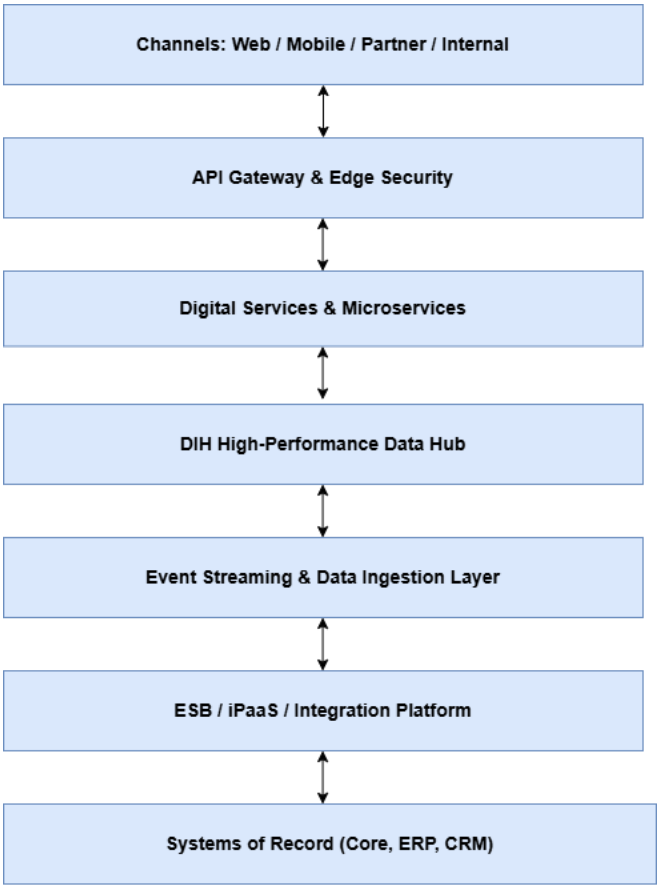
3. Cross-cutting concerns

- **Security & Identity:** IAM, SSO, OAuth2/OIDC, fine-grained authorization, secrets management.
- **Governance & Catalog:** API catalog, data catalog, lineage, ownership, SLAs.
- **Observability:** Centralized logging, metrics, tracing, health checks, SLOs.
- **Resilience:** Circuit breakers, bulkheads, retries, backpressure, graceful degradation.
- **Data Management:** Data quality, PII handling, retention, compliance (GDPR, etc.).

4. Canonical interaction flow

1. **Channel calls API** via API gateway.
2. **Gateway routes** to a domain microservice.
3. **Microservice reads** from the DIH data store (materialized view).
4. If needed, **microservice enriches** with real-time SoR call via integration layer.
5. **Response returned** to channel with low latency.
6. **SoR changes** (transactions, updates) are captured via CDC or events.
7. **Event streaming layer** publishes change events.
8. **Ingestion services** update the DIH data store/materialized views.

5. Simple canonical diagram



Step-by-Step DIH Blueprint

1. Define scope and value cases

- Identify priority journeys:
  - Examples: Customer 360, product catalog, order status, pricing, offers.
  - Focus on read-heavy, latency-sensitive use cases where SoR cannot keep up.

- Define non-functional requirements:
    - Latency: e.g.,
    - Throughput: peak requests per second.
    - Freshness: acceptable data staleness (seconds/minutes).
- Map systems of record (SoR):
- For each journey, list authoritative systems and integration style (API, DB, mainframe, files).

2. Design the logical architecture

- Confirm layered model:
    - Channels → API Layer → Microservices → DIH Data Hub → Event/Ingestion → Integration → SoR.
- Define domain boundaries:
    - Group capabilities into domains (Customer, Product, Order, Payments).
    - Assign domain ownership (teams, product owners).
- Choose integration styles per SoR:
    - Real-time: APIs, messaging.
    - Near real-time: CDC from DB logs.
    - Batch: ETL for non-critical or legacy data.

3. Select core DIH technologies

- **High-performance data store:**
    - Options: **in-memory grid, distributed cache, or fast NoSQL**.
    - Decide:
        - **Data model:** key-value, document, or tabular.

        - **Sharding & replication:** how to scale and ensure HA.
- **Event streaming backbone:**
    - Choose Kafka/Pulsar (or equivalent).
    - Define:
        - **Topic taxonomy:** by domain and event type.
        - **Retention & compaction:** for replay and rebuild.
- **API and microservices stack:**
    - API gateway (rate limiting, auth, routing).
    - Microservices framework (language, runtime, container/orchestration).
- **Integration platform:**
    - ESB/iPaaS or custom adapters.
    - CDC tooling (e.g., log-based connectors).

4. Model data and views for the hub

- **Define canonical data models:**
    - For each domain, define **canonical entities** (Customer, Product, Order).
    - Specify **mandatory attributes** and **PII classification**.
- **Design materialized views:**
    - Start from **channel use cases**, not SoR schemas.
    - Examples:
    - **Customer 360 view:** identity, contact, accounts, preferences.
    - **Product view:** price, availability, attributes, promotions.
- **Set freshness and TTL policies:**

- For each view:
  - **TTL:** how long data can live in the hub.
  - **Refresh strategy:** event-driven, scheduled, or on-demand.

5. Design ingestion and synchronization

- **Event and CDC flows:**
  - For each SoR:
    - Decide **which events** to publish (created/updated/deleted).
    - Configure **CDC connectors** where possible.
- **Ingestion services:**
  - Implement services that:
    - **Consume events**, transform and enrich them.
    - **Write to DIH views** in the high-performance store.
  - Include **idempotency** and **dead-letter handling**.
- **Rebuild strategy:**
  - Define how to **rebuild views**:
    - From **event history** (preferred).
    - From **full SoR extracts** if needed.

6. Implement digital services and APIs

- **Domain microservices:**
  - Each microservice:
    - Owns a **domain capability**.
    - Reads primarily from the **DIH data hub**.

        - Calls SoR **only when necessary** (e.g., for transactions).
- **API contracts:**
    - Design **versioned APIs** aligned with domain models.
    - Include **pagination, filtering, and partial responses** for performance.
- **Fallback and enrichment:**
    - Define when a service:
        - **Serves from DIH only**.
        - **Enriches from SoR** (e.g., latest balance).
        - **Fails gracefully** if SoR is unavailable.

7. Cross-cutting concerns and governance

- **Security and privacy:**
    - OAuth2/OIDC, fine-grained authorization.
    - Pseudonymization or encryption for PII in the hub.
    - Data retention and right-to-be-forgotten processes.
- **Observability:**
    - Tracing across **channel → API → microservice → DIH → SoR**.
    - Metrics: cache hit ratio, event lag, view rebuild time, error rates.
- **Data and API governance:**
    - **API catalog** and **data catalog** with ownership and SLAs.
    - Change management for **schemas and contracts**.

8. Incremental rollout plan

- **Start with one flagship journey:**

- Example: **“Customer 360 for mobile app”**.
- Implement end-to-end:
  - Events/CDC → DIH view → microservice → API → channel.

- **Measure and harden:**
  - Validate:
    - Latency, throughput, freshness.
    - Operational runbooks and incident handling.
- **Scale to additional domains:**
  - Reuse:
    - **Event patterns**, **view patterns**, **microservice templates**, **governance model**.
  - Add new domains (Product, Order, etc.) with minimal architectural change.

DIH Capability Model for Enterprise Architecture Deliverables

1. Channel & Experience Enablement

Capabilities that expose consistent, secure, high-performance interfaces to digital consumers.

1.1 API Exposure & Experience

- API productization & lifecycle management
- API versioning & backward compatibility
- Developer portal & onboarding
- Traffic management & throttling

1.2 Edge Security

- Authentication & authorization (OAuth2/OIDC/JWT)
- Threat protection (WAF, bot mitigation)
- Rate limiting & quota enforcement

2. Digital Services & Domain Orchestration

Capabilities that encapsulate business logic and orchestrate data from the DIH and SoR.

2.1 Domain Microservices

- Domain-aligned service boundaries
- Stateless compute & horizontal scaling
- Domain-logic encapsulation

2.2 Service Orchestration

- Composition of multiple domain services
- Synchronous and asynchronous orchestration
- Policy enforcement (consent, entitlements, masking)

2.3 API Contract Governance

- Schema definition & validation
- API cataloging & discoverability
- Contract testing & change governance

3. High-Performance Data Hub

The core of the DIH: low-latency, channel-optimized data access.

3.1 Data Modeling & View Management

- Canonical domain models
- Materialized view creation & lifecycle
- View versioning & schema evolution

3.2 High-Performance Storage

- Distributed in-memory caching
- NoSQL/document/key-value storage
- Sharding, replication, and HA

3.3 Data Freshness & Consistency

- TTL policies & staleness rules
- Cache invalidation strategies
- Read/write consistency models

4. Event Streaming & Data Ingestion

Capabilities that keep the DIH synchronized with systems of record.

4.1 Event Backbone

- Topic taxonomy & naming conventions
- Event retention & compaction
- Replay & recovery

4.2 Change Data Capture (CDC)

- Log-based CDC ingestion
- Schema change detection
- Idempotent event generation

4.3 Ingestion & Transformation Services

- Event enrichment & normalization
- Deduplication & ordering
- Error handling & dead-letter queues

5. Enterprise Integration Layer

Bridges the DIH with legacy and modern SoR.

5.1 Integration Connectors

- API, SOAP, file, batch, MQ, mainframe adapters
- iPaaS/ESB integration flows

5.2 Real-Time & Batch Integration

- Real-time SoR lookups
- Scheduled ETL/ELT jobs
- Hybrid integration patterns

5.3 Transactional Integrity

- Compensating transactions
- Saga patterns for distributed consistency

6. Data Governance & Compliance

Ensures trust, lineage, and regulatory compliance.

6.1 Data Catalog & Metadata

- Data lineage & provenance
- Business glossary & semantic definitions

6.2 Data Quality & Validation

- Profiling & quality rules
- PII classification & handling

6.3 Compliance & Retention

- GDPR/CCPA compliance
- Retention & deletion policies
- Consent management

7. Security, Identity & Access

Cross-cutting security capabilities.

7.1 Identity & Access Management

- Role-based and attribute-based access
- Secrets & key management

7.2 Data Security

- Encryption in transit & at rest
- Tokenization & pseudonymization

7.3 Threat Detection

- Anomaly detection
- Security event monitoring

8. Observability & Operations

Ensures the DIH is operable, observable, and resilient.

8.1 Monitoring & Telemetry

- Metrics, logs, traces
- Distributed tracing across DIH and SoRs

8.2 Reliability Engineering

- Circuit breakers & bulkheads
- Auto-scaling & self-healing
- Chaos testing

8.3 Operational Tooling

- Runbooks & incident response

- Deployment automation & CI/CD
- Configuration & secret rotation

9. Platform & Infrastructure Foundation

The underlying platform capabilities required to run the DIH.

9.1 Runtime Platform

- Container orchestration
- Service mesh
- Multi-region deployment

9.2 Data Platform

- Storage provisioning
- Backup & restore
- Disaster recovery

9.3 Cost & Capacity Management

- Capacity planning
- Cost allocation & chargeback
- Autoscaling policies

10. Governance & Operating Model

The organizational capabilities that make the DIH sustainable.

10.1 Domain Ownership

- Domain teams & product owners
- RACI for data, APIs, and events

10.2 Change & Release Governance

- API and schema change management

- Event versioning governance

10.3 Portfolio & Roadmap Management

- Domain onboarding roadmap
- Prioritization framework
- Value realization tracking

Summary

DIH Capability Domains:

1. Channel & Experience Enablement
2. Digital Services & Orchestration
3. High-Performance Data Hub
4. Event Streaming & Ingestion
5. Enterprise Integration
6. Data Governance & Compliance
7. Security & Identity
8. Observability & Operations
9. Platform & Infrastructure
10. Governance & Operating Model

This is the canonical capability map you can use in:

- Target architecture documents
- Operating model definitions
- RFPs and vendor evaluations
- Maturity assessments
- Roadmaps and sequencing plan

DIH vs API Gateway vs ESB Comparison

Purpose of the Comparison

Clients often confuse these three because all of them "sit in the middle" of an enterprise architecture.
This model clarifies **what each one does**, **what problem it solves**, and **why they are not interchangeable**.

1. One-Sentence Definitions

- **Digital Integration Hub (DIH):**
  A **high-performance data and integration layer** that decouples digital channels from slow or legacy systems using **cached/materialized views + event-driven sync**.
- **API Gateway:**
  A **traffic manager and security front door** for APIs; it exposes, protects, and routes API calls but does **not integrate or store data**.
- **Enterprise Service Bus (ESB):**
  A **centralized integration middleware** that orchestrates and transforms messages between systems, typically in a **synchronous, request/response** model.

2. Core Purpose Comparison

Capability	DIH	API Gateway	ESB
Primary Purpose	Decouple channels from SORs using fast data	Securely expose and manage APIs	Integrate and orchestrate enterprise systems
Optimized for	High-volume, low-latency digital workloads	API traffic control and security	System-to-system integration
Data storage	Yes (cached/materialized views)	No	No
Event-driven	Native	Minimal	Possible, not native
Orchestration	Light (via microservices)	No	Strong

3. Architectural Role

Digital Integration Hub (DIH)

- Sits **between microservices and systems of record**
- Provides **fast, aggregated, channel-ready data**
- Uses **CDC + events** to stay in sync
- Reduces load on legacy systems

Think: "High-speed digital data layer."

API Gateway

- Sits **at the edge**
- Protects APIs, enforces policies, routes traffic
- Does **not** integrate, transform, or store data

Think: "Secure front door for APIs."

ESB

- Sits **in the integration layer**
- Handles **protocol mediation, transformation, routing**
- Often synchronous and tightly coupled

Think: "Legacy integration backbone."

4. Strength & Weaknesses

	DIH	API Gateway	ESB
Strength	Ultra-low latency, scalable, channel-optimized data; decoupled SoRs	Security, throttling, routing, API lifecycle	Mature integration patterns, transformation, orchestration

Weaknesses	Requires event/CDC maturity; new data governance	No integration or data capabilities	Can become a bottleneck; not built for digital scale
Best for	Mobile/web scale, omnichannel, modernization	API exposure, partner integration	Legacy system integration, synchronous flow

5. When to Use Each

Use a DIH when:

- Digital channels overload your SoRs
- You need **sub-100ms** responses
- You want to **decouple** channels from legacy systems
- You need **materialized views** (Customer 360, product catalog, etc.)

Use an API Gateway when:

- You need to expose APIs securely
- You want to manage traffic, quotas, keys, tokens
- You need a developer portal or partner onboarding

Use an ESB when:

- You must integrate legacy systems
- You need message transformation or protocol mediation
- You have synchronous, system-to-system workflows

6. How They Work Together

A modern architecture uses **all three**, but for different jobs:

- **API Gateway** is the **front door**
- **DIH** is the **fast digital brain**

- **ESB** is the **legacy translator**

Flow example:

1. Mobile app calls API → API Gateway
2. Gateway routes to microservice
3. Microservice reads from DIH (fast, aggregated data)
4. DIH stays updated via ESB/CDC/events from SoRs

This is the **canonical Gartner pattern** for digital modernization.

7. One-Page Summary (Board-Ready)

Layer	DIH	API Gateway	ESB
Edge	✗	✓	✗
Digital/API Services	✓	✓ (routing only)	✓
High-Performance Data	✓	✗	✗
Integration	✓ (event-driven)	✗	✓
Legacy Connectivity	✓	✗	✓

Maturity Model – DIH vs API Gateway vs ESB Adoption

This model compares the three technologies across **five maturity levels**, showing how organizations typically evolve their integration and digital-enablement capabilities.

It highlights what "good" looks like, what changes at each stage, and how each technology contributes to modernization.

1. Maturity Levels Overview

Maturity Level	Description
L1 – Ad Hoc	Point-to-point integrations, siloed APIs, limited governance

L2 – Structured	API gateway adoption, basic ESB patterns, early standardization
L3 – Integrated	ESB becomes backbone, API gateway formalized, early caching
L4 – Digital-Ready	DIH introduced to decouple channels, event-driven patterns emerge
L5 – Modernized & Scalable	Full DIH adoption, event-driven enterprise, legacy minimized

2. Capability Maturity Comparison

A – Integration Backbone (ESB)

Maturity Level	ESB Characteristics
L1 – Ad Hoc	Point-to-point integration; no ESB
L2 – Structured	ESB introduced for basic routing and transformation
L3 – Integrated	ESB becomes central integration hub; heavy orchestration
L4 – Digital-Ready	ESB still used, but offloading from high-volume digital traffic
L5 – Modernized	ESB minimized; event streaming and microservices replace synchronous flows

Narrative:

ESB is strongest in L2 – L3 but becomes a bottleneck at scale. Modern architecture reduces ESB reliance as DIH and event streaming take over.

B – API Exposes & Management (API Gateway)

Maturity Level	API Gateway Characteristics
L1 – Ad Hoc	APIs exposes directly from systems; no gateway
L2 – Structured	API gateway introduced for routing and security
L3 – Integrated	API lifecycle management, developer portal, versioning
L4 – Digital-Ready	Gateway fronts microservices + DIH; traffic offloaded from SORs
L5 – Modernized	API gateway becomes the unified digital front door for all channels

Narrative:

API gateways mature early and remain essential throughout all levels. They do not replace ESB or DIH – they complement them.

C – Digital Data Layer (DIH)

Maturity Level	DIH Characteristics
L1 – Ad Hoc	No caching; channels hit SoRs directly
L2 – Structured	Basic caching (CDN, in-app cache)
L3 – Integrated	Tactical caching; still SoR-dependent; no event synch
L4 – Digital-Ready	DIH introduced; materialized views, CDC, event-driven synch
L5 – Modernized	DIH becomes primary read layer; SoR load minimized; omnichannel consistency achieved

Narrative:

DIH only appears at higher maturity levels because it requires event streaming, CDC, and domain modeling. It is the key enabler of digital scale and legacy decoupling.

3. Cross-Technology Comparison by Maturity Level

Level 1 – Ad Hoc

- ESB: None
- API Gateway: None
- DIH: None
- Symptoms: Slow channels, duplicated integrations, fragile systems

Level 2 – Structured

- ESB: Introduced for basic integration
- API Gateway: Introduced for API exposure

- DIH: Not present
- Outcome: Integration becomes manageable but not scalable

Level 3 – Integrated

- ESB: Central orchestration engine
- API Gateway: Full API lifecycle management
- DIH: Tactical caching only
- Outcome: Good internal integration, but digital channels still overload SoRs

Level 4 – Digital-Ready

- ESB: Still used, but offloaded from digital workloads
- API Gateway: Fronts microservices + DIH
- DIH: Introduced as high-performance digital data layer
- Outcome: Channels decoupled from SoRs; latency and scale improve dramatically

Level 5 – Modernized & Scalable

- ESB: Reduced footprint; event-driven patterns dominate
- API Gateway: Unified digital front door
- DIH: Primary read layer for all channels; event-driven enterprise
- Outcome: True digital agility; legacy systems insulated from channel demand

4. Summary

- ESB: solves system-to-system integration
- API Gateway: solves secure API exposure
- DIH: solves digital scale, latency, and legacy decoupling

They are **not competing technologies**. They represent **different maturity stages** and **different architectural responsibilities**.

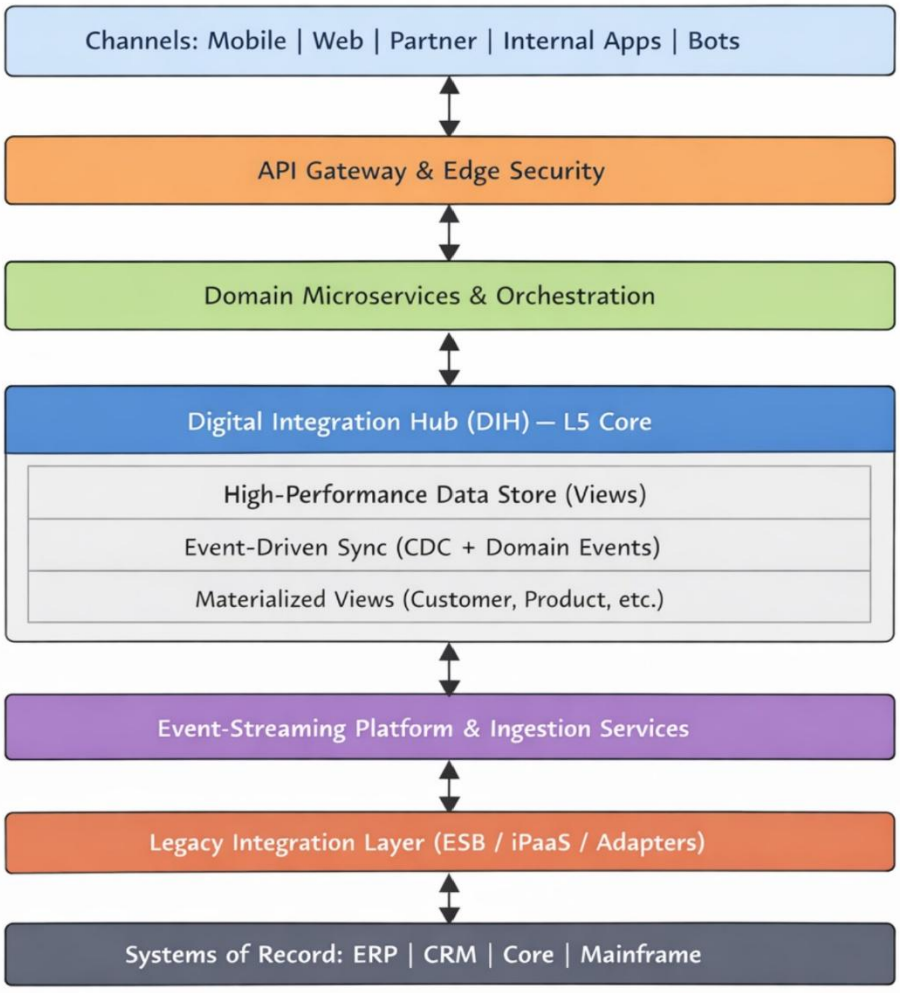
L5 Target Architecture – Digital Integration Hub-Centric Enterprise

At Level 5, the enterprise has transitioned from a legacy-centric integration model to a **digital-first, event-driven, API-productized architecture**.
The DIH becomes the **primary read layer**, the API Gateway becomes the **unified digital front door**, and the ESB is minimized to **legacy connectivity only**.

This architecture supports:

- Omnichannel digital experiences
- Real-time personalization
- High-volume, low-latency workloads
- Legacy modernization without rewriting core systems
- Event-driven enterprise operations

1 Architecture Overview (Layered Model)



Cross-cutting: Security, Governance, Observability, Data Quality, Platform Ops

2 Core Architecture Domains

2.1 Channel & Experience Layer

Capabilities:

- Unified API access for all channels

- Consistent authentication (OAuth2/OIDC)
- Rate limiting, quotas, threat protection
- Developer portal for partners and internal teams

Outcome:
Channels never call systems of record directly.

2.2 API Gateway & Edge Security

Capabilities:

- API productization
- Policy enforcement
- Routing to microservices and DIH
- Zero-trust edge security

Outcome:
The gateway becomes the **single digital entry point**.

2.3 Domain Microservices Layer

Capabilities:

- Domain-aligned microservices (Customer, Product, Order, Payments)
- Stateless compute, autoscaling, service mesh
- Orchestration of DIH + SoR calls
- Event publishing and consumption

Outcome:
Business logic is modular, scalable, and independently deployable.

2.4 Digital Integration Hub (L5 Core)

This is the **heart** of the target architecture.

Capabilities:

- High-performance distributed data store
- Materialized views optimized for digital journeys
- Event-driven synchronization from SoR
- TTL, staleness rules, partial updates
- Schema governance and versioning
- Replay and rebuild from event logs

Outcome:
DIH becomes the **primary read layer** for all digital channels.

2.5 Event Streaming & Data Ingestion

Capabilities:

- Enterprise event backbone (Kafka/Pulsar)
- Domain event taxonomy
- CDC ingestion from RDBMS and mainframe
- Enrichment and transformation services
- Dead-letter queues, replay, compaction

Outcome:
The enterprise becomes **event-driven**, not request/response-driven.

2.6 Legacy Integration Layer (ESB / iPaaS)

Capabilities:

- Protocol mediation (SOAP, MQ, file, batch)
- Legacy adapters for mainframe, ERP, CRM
- Transactional orchestration where required

Outcome:
ESB is **minimized** to legacy connectivity only.

2.7 Systems of Record

Capabilities:

- Authoritative data and transactional integrity
- Exposed via events, CDC, or synchronous APIs

Outcome:
SoR are insulated from digital load.

3 Cross-Cutting Architecture Domains

3.1 Security & Identify

- Zero-trust architecture
- Fine-grained authorization (ABAC/RBAC)
- Secrets and key management
- PII masking and tokenization

3.2 Data Governance

- Data catalog and lineage
- Canonical domain models
- Schema evolution governance
- Data quality and validation

3.3 Observability & Operations

- Distributed tracing across all layers
- Metrics: event lag, cache hit ratio, API latency

- Automated runbooks and incident response
- Chaos engineering for resilience

3.4 Platform & Infrastructure

- Container orchestration (Kubernetes)
- Service mesh (mTLS, traffic shaping)
- Multi-region deployment
- Automated CI/CD pipelines

4 End-to-End Request Flow (L5)

Digital Read Flow

1. Mobile app calls API Gateway
2. Gateway routes to domain microservice
3. Microservice reads from DIH (materialized view)
4. DIH returns low-latency, aggregated data
5. Response returned to channel in <100ms

Data Sync Flow

1. SoR emits change (CDC or domain event)
2. Event streaming platform publishes event
3. Ingestion service enriches and transforms
4. DIH updates materialized view
5. Microservices and channels see updated data instantly

5 Target-State Component Map (Vendor-Agnostic)

Compute & Services

- Kubernetes or equivalent

- Service mesh
- Stateless microservices

API & Edge

- API Gateway
- WAF
- Identity provider (OIDC/OAuth2)

DIH Core

- Distributed in-memory data grid
- NoSQL/document store
- Materialized view manager
- TTL and staleness engine

Event & Data

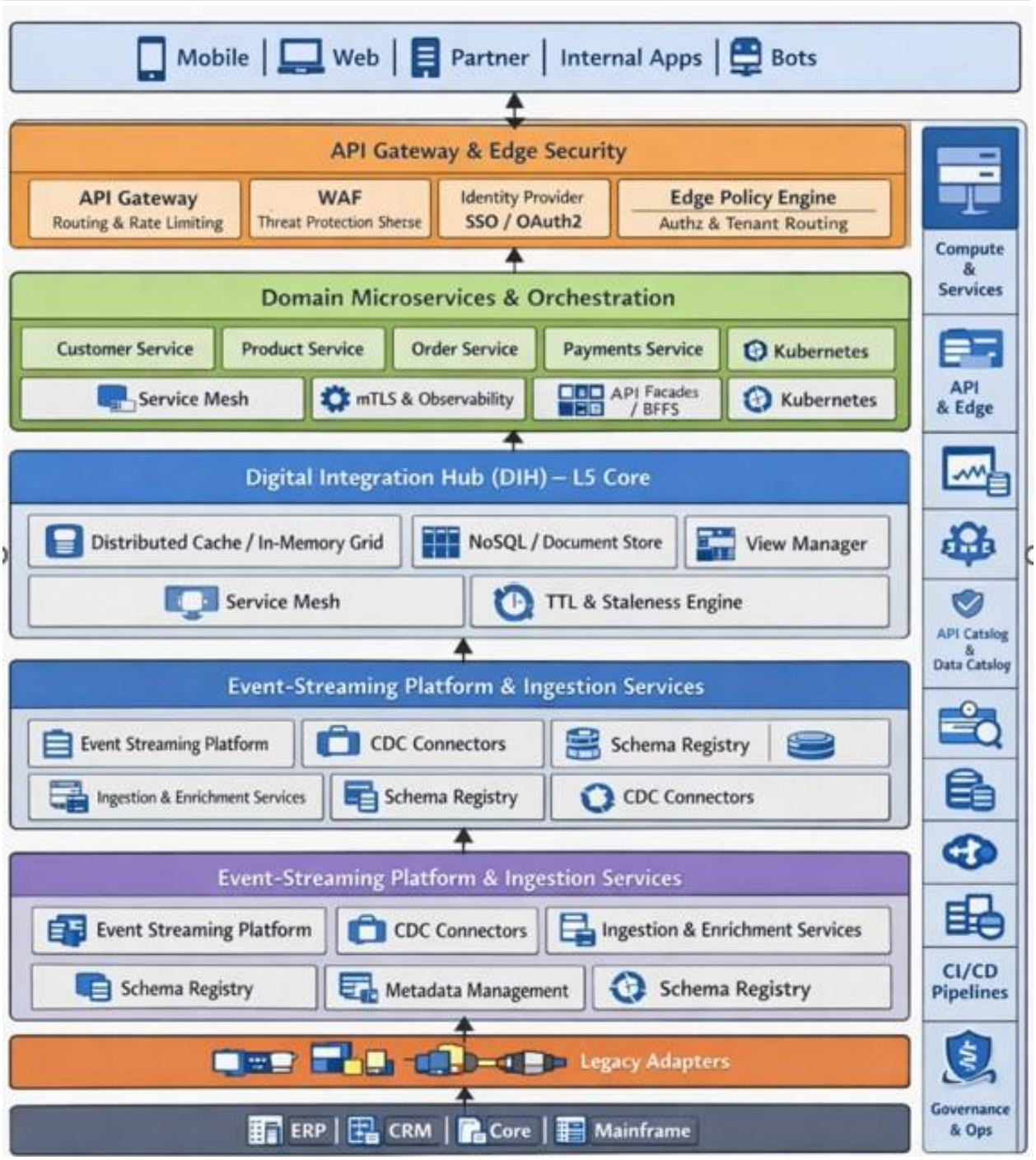
- Event streaming platform
- CDC connectors
- Ingestion and enrichment services
- Schema registry

Integration

- ESB/iPaaS for legacy
- Adapters for mainframe, ERP, CRM

Governance & Ops

- API catalog
- Data catalog
- Observability stack
- CI/CD pipelines



Channels (Mobile | Web | Partner | Internal Apps | Bots)

This layer represents all **consumer-facing and internal-facing digital touchpoints**. At L5 maturity, channels are fully decoupled from systems of record and rely exclusively on **API Gateway + DIH** for data and interactions.

Internal Components

Mobile Applications

Native or hybrid mobile apps.

- High-frequency, low-latency interactions
- Personalized experiences
- Token-based authentication (OIDC/OAuth2)

Role:
Primary consumer channel requiring sub-100ms responses.

Web Applications

Browser-based digital experiences.

- Responsive web apps (SPA, PWA)
- High concurrency
- API-driven UI composition

Role:
Unified digital front-end for customers and employees.

Partner Integrations

External B2B or B2B2C channels.

- Partner APIs
- Contract-based access and quotas
- Strong authentication and rate limiting

Role:
Enables ecosystem expansion without exposing internal systems.

Internal Applications

Employee-facing tools and portals.

- Operational dashboards
- Case management
- Internal APIs and microservices

Role:
Consume the same APIs and DIH data as external channels.

Bots & Automation Channels

Conversational and automated interfaces.

- Chatbots, voice assistants, RPA bots
- Event-driven triggers
- API-based orchestration

Role:
Enable automation and conversational experiences.

Outcome at L5 Maturity

- All channels consume **the same APIs and DIH data**
- No channel calls systems of record directly
- Consistent, low-latency omnichannel experiences
- Rapid rollout of new digital products

API Gateway & Edge Security layer

- API Gateway:
  Routing, rate limiting, quotas, API keys, request/response shaping.

- Web Application Firewall (WAF):
  Threat protection, OWASP rules, bot mitigation, geo/IP filtering.
- Identity Provider (OIDC/OAuth2):
  Token issuance, SSO, federation, JWT validation, refresh flows.
- Edge Policy Engine:
  Centralized policy enforcement (authZ, headers, tenant routing).
- Developer Portal:
  API documentation, onboarding, client registration, sandbox keys.

Domain Microservices & Orchestration layer

- Domain Microservices:
  Customer, Product, Order, Payments, etc.—each as its own service box.
- Service Mesh:
  mTLS, traffic routing, retries, circuit breaking, observability between services.
- Container Orchestration (e.g., Kubernetes):
  Scheduling, scaling, rolling deployments, health checks.
- API Facades / BFFs (optional):
  Channel-specific facades (Mobile BFF, Web BFF) that orchestrate multiple domain services.
- Saga / Orchestration Components:
  Where needed for long-running, cross-domain workflows.

Digital Integration Hub (DIH) – L5 Core

This layer is the **central digital data platform**—decoupling digital channels from systems of record by providing **low-latency, curated, channel-optimized views**. It is not a cache, but a governed, event-synchronized, high-performance read layer.

Internal Components:

- **Distributed Cache / In-Memory Grid**
  - Horizontally scalable, low-latency data access
  - Co-located compute for real-time enrichment
  - Examples: Apache Ignite, Hazelcast, Redis Enterprise
- **NoSQL / Document Store**
  - Persistent storage for materialized views
  - Supports flexible schemas and high write throughput
  - Examples: MongoDB, Couchbase, DynamoDB
- **Materialized View Manager**
  - Pre-computed, denormalized views for digital journeys
  - View lifecycle management, schema versioning
  - Examples: Customer 360, Product Catalog, Order Summary
- **TTL & Staleness Engine**
  - Time-to-live policies per view
  - Staleness thresholds and refresh strategies
  - Event-driven invalidation and partial updates

Event-Streaming Platform & Ingestion Services

This layer is the **enterprise nervous system** that moves data from systems of record into the DIH and across domains in real time. It enables **event-driven synchronization**, **loose coupling**, and **high-throughput data distribution** across the digital ecosystem.

It is foundational for achieving L5 maturity because it ensures that all downstream systems receive timely, consistent, and repayable data.

Internal Components

Event Streaming Platform

The backbone for real-time, high-volume event distribution.

- Durable, partitioned event logs
- High-throughput publish/subscribe
- Event retention, compaction, and replay
- Domain event taxonomy and governance
- Examples: Kafka, Pulsar, Redpanda

Role:

Ensures every change in the enterprise can be broadcast, consumed, and replayed reliably.

CDC Connectors (Change Data Capture)

Bridges legacy systems and modern event streams.

- Log-based CDC from RDBMS and mainframe
- Schema change detection
- Idempotent event generation
- Debezium-style connectors or vendor equivalents

Role:

Turns SoR changes into real-time events without modifying legacy applications.

Ingestion & Enrichment Services

Transforms raw events into DIH-ready data.

- Event enrichment (lookups, joins, metadata injection)
- Normalization and mapping to canonical models
- Deduplication and ordering guarantees
- Error handling and dead-letter queues

- Stateless or stateful stream processing

Role:

Prepares data for DIH materialized views and downstream consumers.

Schema Registry

Centralized governance for event and view schemas.

- Versioned schemas for events and CDC payloads
- Backward/forward compatibility rules
- Validation and enforcement at publish/consume time
- Supports Avro, JSON Schema, Protobuf

Role:

Prevents schema drift and ensures safe evolution of event contracts.

Legacy Integration Layer (ESB / iPaaS / Adapters)

This layer provides the **connective tissue** between modern digital platforms and legacy systems of record. At L5 maturity, it no longer acts as the enterprise's central integration backbone — instead, it becomes a **specialized compatibility layer** that handles protocol mediation, legacy orchestration, and system-specific adapters.

Its purpose is to **minimize change in systems of record** while enabling the enterprise to modernize around them.

Internal Components

Enterprise ESB / iPaaS

The remaining orchestration and mediation engine for legacy systems.

- Protocol mediation (SOAP, MQ, FTP, file drops)
- Message transformation (XML ↔ JSON, EDI, proprietary formats)

- Synchronous request/response orchestration
- Transactional integrity and compensating transactions
- Routing, mapping, and enrichment for legacy endpoints

Role:
Handles integration patterns that cannot be moved to event-driven or microservices architectures due to system constraints.

Legacy Adapters (ERP, CRM, Core, Mainframe)

Purpose-built connectors for systems that cannot expose modern APIs or events.

- Mainframe connectors (CICS, IMS, COBOL copybooks)
- ERP adapters (SAP IDoc/BAPI, Oracle EBS, Microsoft Dynamics)
- CRM adapters (Siebel, Salesforce legacy integrations)
- MQ-based or file-based integration endpoints
- Batch extract/transform/load (ETL) pipelines where required

Role:
Provides stable, governed access to systems of record without requiring invasive changes to those systems.

Batch & File Integration Services

Still required for systems that operate on scheduled cycles.

- Nightly or hourly batch jobs
- File ingestion and distribution (SFTP, shared drives, secure file gateways)
- Bulk data movement for reconciliation, reporting, or compliance
- Error handling and restartability

Role:
Supports SoR that cannot operate in real time but must still participate in enterprise data flows.

Synchronous SoR Access Services

Used when real-time reads/writes are unavoidable.

- Low-latency adapters for transactional SoR calls
- Connection pooling and throttling
- Circuit breakers and fallback logic
- Request/response mapping to canonical models

Role:
Allows microservices to call SoR safely when DIH cannot serve the request (e.g., real-time balance checks).

Outcome at L5 Maturity

- ESB/iPaaS footprint is **reduced**, not eliminated
- Legacy systems remain stable and unchanged
- Modern systems interact via **events and DIH**, not ESB
- Legacy integration becomes **predictable, governed, and isolated**
- The enterprise avoids risky "big bang" modernization of core systems

Systems of Record (ERP | CRM | Core | Mainframe)

This layer contains the **authoritative systems** that hold the enterprise's core transactional and master data. At L5 maturity, these systems remain stable and unchanged, while the rest of the architecture modernizes around them.

Internal Components

ERP Systems

Enterprise resource planning platforms.

- Finance, procurement, supply chain
- IDoc/BAPI interfaces
- Batch and real-time integration endpoints

Role:
Authoritative source for financial and operational data.

CRM Systems

Customer relationship management platforms.

- Customer master data
- Case management, sales, service
- Legacy SOAP or MQ interfaces

Role:
Authoritative source for customer interactions and profiles.

Core Line-of-Business Systems

Industry-specific transactional systems.

- Core banking, policy admin, billing, claims, order management
- Often monolithic and tightly coupled
- Limited API or event capabilities

Role:
Authoritative source for mission-critical transactions.

Mainframe Systems

Long-lived, high-reliability systems.

- COBOL programs, CICS transactions
- VSAM files, DB2 databases
- Batch windows and nightly cycles

Role:
Authoritative source for high-volume, high-integrity workloads.

Outcome at L5 Maturity

- Systems of record remain stable and unchanged
- DIH and event streaming absorb digital load
- Legacy systems are insulated from channel traffic
- Modernization occurs around, not inside, the SoR

Governance & Ops

This layer provides the **operational backbone and governance controls** that ensure the entire digital ecosystem is secure, observable, compliant, and continuously deployable. At L5 maturity, Governance & Ops is **fully integrated**, **automated**, and **policy-driven**, enabling safe, rapid change across all layers.

Internal Components

API Catalog & Data Catalog

Centralized discovery and governance for APIs and data assets.

- API inventory, ownership, lifecycle states
- Data lineage, metadata, and semantic definitions
- Schema versioning and compatibility rules
- SLA/SLO tracking for APIs and data products

Role:
Ensures transparency, reuse, and governance across all integration and data assets.

Observability Stack

Unified monitoring across microservices, DIH, events, and legacy systems.

- Distributed tracing (end-to-end request flows)
- Metrics (latency, throughput, error rates, event lag)
- Centralized logging
- Dashboards and alerting

Role:
Provides real-time visibility and proactive issue detection across the entire architecture.

Secret & Key Management

Secure handling of credentials, tokens, certificates, and encryption keys.

- Vault-based secret storage
- Automated rotation
- mTLS certificate management
- Encryption key lifecycle governance

Role:
Protects sensitive assets and enforces zero-trust principles.

CI/CD Pipelines

Automated delivery pipelines for all components.

- Build, test, and deployment automation
- Canary and blue/green deployments
- Infrastructure-as-code (IaC)
- Policy-as-code for compliance gates

Role:
Enables safe, rapid, repeatable deployments across all layers.

Outcome at L5 Maturity

- Full observability across all layers
- Automated, policy-driven deployments
- Strong governance for APIs, data, and events
- Zero-trust security posture
- Operational resilience and rapid recovery

6 What "Good" Looks Like at L5

Digital

- Sub-100ms response times
- Omnichannel consistency
- Zero downtime deployments

Data

- DIH is the primary read layer
- Event-driven sync across domains
- Full lineage and governance

Integration

- ESB footprint reduced by 60–90%
- Event streaming replaces synchronous flows

Architecture

- Domain-driven microservices
- API-first productization
- Multi-region resilience

7 L5 Target Architecture Summary

The L5 enterprise is:

- **API-first**
- **Event-driven**
- **DIH-centric**
- **Legacy-decoupled**
- **Omnichannel-consistent**
- **Scalable and resilient**

This is the **north star** for any organization modernizing digital channels without rewriting core systems.

Version 1.0

Author: Christian Kobsa, Strategic Enterprise Architect
Digital Enterprise Architecture Advisors LLC

About DEAA

Digital Enterprise Architecture Advisors (DEAA) provides strategic architecture, modernization guidance, and capability modeling for enterprises operating across the U.S. and Europe. DEAA specializes in helping organizations design scalable digital platforms, rationalize complex technology landscapes, and build modular, future-ready operating models. Our work blends consulting-grade rigor with practical, actionable architectural patterns that accelerate transformation and reduce complexity.