naehas

Real-Time Composable Customer Data Platform for Financial Services

A Reference Architecture for IT Leaders.

Executive Summary

Modern financial institutions need customer-safe, real-time systems that can **detect intent, decide, deliver, and fulfill**—without brittle glue code or stale data. This paper describes a **composable, event-driven architecture** that powers offer fulfillment today and extends to journey orchestration, a customer data platform (CDP), decisioning, and **lays the groundwork for AI agents**—all on the same shared services. The result is **millisecond responsiveness**, **consistent data**, and **end-to-end observability** across every customer interaction.

The approach pairs a **financial-grade semantic model** with **modular content** and a **design studio** to deliver compliant, personalized experiences powered by real-time intelligent data. Services are **composable**—apps share the same data, decisioning, and governance fabric—and they **run next to your data** (e.g. read-only on Snowflake via secure shares/PrivateLink, or on Databricks via secure sharing/peering), avoiding another processing engine or copy. Outcome: **consistent, auditable execution** at the speed and scale of modern banking.

> *Uniquely built for financial services.*
> *Compliance and transactional integrity at its core.*
> *Composable by design.*

Capabilities for AI-Ready Financial Institutions

Three Non-Negotiable Capabilities

1 Transactional Integrity and Auditability

Customer decisions must rest on a complete, correctly ordered history—every event processed once, in sequence, with rollback and lineage.

Outcome: Balances, eligibility, disclosures, and fulfillment are correct by construction (not “eventually consistent”), defensible to regulators, and explainable to customers.

2 A Financial-Grade Semantic Data Model

Offers, products, disclosures, modular content, identities/households, journeys, and fulfillment must live as typed business objects that rules and AI agents can reason over.

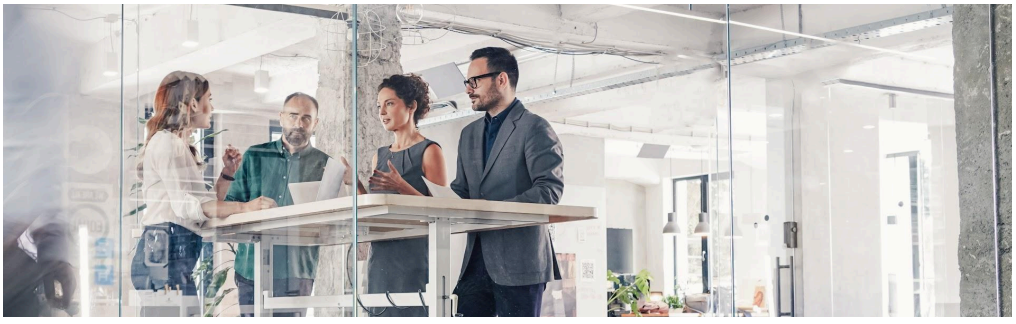
Outcome: Systems only propose what can be fulfilled, bind the right disclosures automatically, and provide human-readable rationales.

Example: For a new product launch, one Product definition with versions, pricing, and eligibility propagates consistent Offers, correct Disclosures, aligned Journeys, and accurate analytics—without channel-by-channel rework.

3 Composable by Design on Your Data Cloud

Run decisioning and activation next to governed data (e.g. Snowflake, Databricks) via secure sharing—zero new copies/engines—with a hot context tier for sub-second lookups.

Outcome: Faster time-to-value, lower risk, elastic scale on existing controls, and real-time experiences that don’t compromise correctness.



Core Platform Capabilities

This isn't a product catalog. It's a capability lens—the minimum set of building blocks financial institutions need to deliver real-time, compliant experiences and to scale new use cases from pilot to production on a single, composable data foundation.

Event Stream Backbone

A durable, scalable bus for all customer and operational events (account updates, transactions, clicks, CRM changes). This is the system's "nervous system" that moves data in real time. Stream customer and operational events so decisions land while the customer is still engaged.

Real-Time Decision Engine

Ultra-low-latency event processing that recognizes complex, time-based patterns and triggers decisions instantly. Think of it as the platform's "brain."

Analytical Context Store

A real-time analytical store ("memory") for sub-second lookups across large volumes of recent and historical data to enrich in-flight decisions.

Unified Object Graph and Lineage (Data Model)

Not one record, but a **governed graph of typed objects**—offers, products, disclosures, campaigns, collateral, journeys, fulfillment events—with explicit relationships and versioned lineage. The system understands impact across the graph (e.g. a product or rate change) and propagates updates safely, enabling **fast, defensible audits**.

Observability, Audit, and Governance

Prove every step with end-to-end tracing from the triggering event through delivery and fulfillment, plus consistent data lineage for analytics and compliance. Trace event → decision → content → offer → fulfillment → outcome; investigate exceptions and optimize safely.

Identity, Policy, and Real-Time Customer Profile (RCP)

Resolves customers, accounts, and households over time and serves low-latency, relationship-aware profiles – enforcing consent, purpose limits, and entitlements at access with full lineage for audit.

So what does a real-time, composable CDP change?

Relevant decisions while the customer is still engaged, **deterministic fulfillment** with reconciliation against governed facts, and **audits answered in hours, not weeks**, on one architecture that is reused for Journeys and Offer Decisioning.

Critical Applications for FinServ Built on a Composable Data Platform

A real-time, composable CDP turns your data cloud into an execution layer for outcomes that matter in banking: relevant experiences in-session, promises fulfilled deterministically, and audits answered in hours—not weeks. The same governed identity, object graph, and policy controls power four critical apps:

Customer Data Platform (CDP) Real-time unification and activation using the unified profile and event backbone.	**Offer Fulfillment** Near-real-time creation, delivery, acceptance, and fulfillment with status tracked centrally.
Journey Orchestration Event-driven next-best-actions across channels using the same decision engine and context store.	**Offer Decisioning** Centralized rules and models that select, bundle, and personalize offers (including complex and nested constructs with variant support).

Composable CDP on Your Data Cloud

Bring decisions to governed data – not data to decisions.

Use a **composable CDP pattern**: keep data where it already lives—in your lake or warehouse—and compose real-time experiences over it via governed, read-only access. The result is less data movement, a smaller operational surface area, and stronger governance, because existing enterprise controls continue to protect the data being read.

Read-Only on Your Data Platform:
Connect via secure sharing/peering to your data cloud (e.g. Snowflake/Databricks) to read governed, production-grade data **without copying** it into another engine.

Push-Down and Cache Smartly:
Eligibility and personalization rules run with sub-second lookups using the Analytical Context Store, while heavier joins are delegated to your data cloud where appropriate—so performance scales with your warehouse.

Operational System(s) of Record:
Offer and profile repositories operate as part of a unified object graph (separate, versioned objects with relationships and lineage). Delivery may use denormalized projections, but the authoritative truth remains in the graph—ensuring consistency across channels and fulfillment without implying one mega-record.

Example: Offer Management and Eligibility on Snowflake (Composable CDP)

1. Events land on the event stream backbone (e.g. funding, logins).
2. Identity and RCP resolve/attach the right customer, account, and household. Resolution happens at access time, so mergers and ID changes don't break activation, fulfillment, or analytics.
3. Eligibility rules evaluate in real time, enriching with read-only facts from Snowflake via secure shares (accounts, transactions, risk flags, etc.).
4. Offer object resides in the Unified Offer Catalog (the operational offer repository within the unified object graph); channels read the same object; fulfillment updates are bound to the same record, down to the version.
5. Tracking and observability ties trigger → decision → delivery → fulfillment for analytics and audit.

Benefit: You get closed-loop speed and consistency without standing up a separate, duplicative processing engine.

End-to-End Offer Lifecycle on a Composable CDP

From design and decision through delivery and acceptance to fulfillment, reconciliation, and learning —a complete lifecycle that drives reliable results and gets smarter every cycle.

The same event-driven, warehouse-native architecture that powers offer fulfillment also supports Journey Orchestration, CDP, and Offer Decisioning—so eligibility logic, content, identity, and governance are shared rather than rebuilt per application.

Model and Decide on a Unified Object Graph

Relationship-Aware Objects:
Offers, Products, Disclosures, Campaigns, Collateral, Journeys, and Fulfillment Events live as separate, versioned objects with explicit relationships and lineage.

Complex/Nested Offers as First-Class:
Hierarchical constructs (bundles, tiers, conditional incentives) are modeled explicitly and evaluated across real-time events and history.

How Eligibility Decides (At a Glance):
Pattern triggers (e.g. checking view → savings view within 10 minutes) plus real-time history checks (e.g. already holds checking) drive precise, compliant decisions.

Closed-Loop Fulfillment on the Same Graph

Ordered, Exactly-Once Processing:
Acceptances and fulfillment events are processed in sequence per customer/account with idempotent keys—1:00 p.m. always precedes 2:00 p.m. for the same stream.

Idempotent Write-Backs to Core/Channels:
Actions (account opening, incentive posting, service changes) execute via policy-aware connectors using correlation IDs and deterministic retries—no duplicates, no drops.

Warehouse-Native Reconciliation (Zero Copy):
Expected outcomes are verified by read-only checks against governed facts in your lake/warehouse (e.g. Snowflake/Databricks via secure sharing/peering); exceptions route to a controlled queue.

Lineage and Impact Analysis:
Changes to any object (e.g. a product rate version) propagate safely: linked offers re-evaluate, disclosures re-bind, collateral is flagged, and full lineage is retained for audit.

Example (Offer Object, Simplified):

```
{
  "offerId": "O-BUNDLE-001",
  "offerName": "New Account Power Bundle",
  "customerId": "C-12345",
  "status": "pending",
  "mainOffer": { "offerType": "newAccount", "product": "Checking Account" },
  "nestedOffers": [
    { "offerType": "preApprovedCreditCard", "product": "Platinum Card" },
    { "offerType": "directDepositBonus", "product": "Savings Account" }
  ]
}
```

For delivery performance, a denormalized offer projection can be read in a single query by channels. The authoritative truth remains separate, versioned objects (offer, product, disclosure, collateral) linked in the unified object graph.

How it Decides (At a Glance):

Event patterns (e.g. “checking view” then “savings view” within 10 minutes) trigger bundle eligibility.

Real-time history checks prevent irrelevant offers (e.g. customer already has checking).

Example: Closed-Loop Fulfillment with Warehouse-Native Access

1. Acceptance captured in a channel (web/mobile/branch/contact center) is emitted on the event backbone in real-time.
2. Identity and Real-Time Customer Profile attaches the correct customer/account/household and links the acceptance to the relevant Offer (versioned) in the Unified Object Graph.
3. Fulfillment action executed via integration (e.g. incentive ledger entry, account attribute update), emitting a Fulfillment Event with an idempotency key.
4. Reconciliation runs by reading governed facts read-only from Snowflake/Databricks (or equivalent) via secure sharing/peering; expected vs. actual is compared. If matched, status transitions to “fulfilled”; if not, an exception/rollback flow replays the affected window with full lineage.
5. Observability stitches trigger + decision + acceptance + action + reconciliation, providing audit-ready evidence and insights for optimization

Benefit: Deterministic, auditable fulfillment with fewer exceptions, faster resolution, and no new engines or data copies—all while preserving ordering guarantees and governed access.

Change Management by Design

Customer and account identifiers change (mergers, temp → permanent IDs, corrections). A mutable architecture keeps history, analytics, and operational apps in sync to avoid split-brain records and customer-visible inconsistencies—while enabling automated, transactional updates across data layers. This prevents ‘split-brain’ profiles and keeps household/role logic intact across systems.

Why It Matters:
Prevents corrupted 360° views, poor risk assessment, and inconsistent experiences when IDs shift; reduces manual, error-prone rework.

To operationalize this, the platform’s Identity and Real-Time Customer Profile service continuously resolves customers, accounts, and households across sources and time – so change never breaks activation, fulfillment, or analytics.

Identity and Real-Time Customer Profile (RCP)

This is the enforcement layer that turns “*change management*” into **consistent execution**. It resolves customer/account/household identities across systems and time, stitches event histories, and applies consent/entitlements as policies at access time—so one **Unified Customer Profile and Offer Catalog** remains accurate for decisioning, delivery, and fulfillment.

What It Does:
Resolves identities across sources; tracks cross-IDs over time; householding and relationship mapping; stitches event trails; enforces consent/entitlements at access time.

How It Helps:
Keeps CX, analytics, and operations in sync when IDs merge/split; prevents duplicate or conflicting offers; aligns fulfillment and disclosures to the right party.

Why It Matters:
Banking data changes constantly (mergers, remediation, data quality fixes). Without a mutable, governed identity layer, you risk “broken promises,” mis-fulfillment, and audit gaps.

The Real-time Customer Profile is key to addressing common challenges in financial services as it provides:

Accuracy under Change:
Prevents “split-brain” profiles during mergers, remediation, or ID corrections.

Household/Relationship-Aware:
Supports one-to-many and many-to-many party/product relationships common to banking.

Privacy by Design:
Applies consent, purpose limits, and entitlements at read time as well as write time.

Feeds Offer Fulfillment: Correct party resolution → accurate eligibility, acceptance, and fulfillment on the correct linked objects in the unified graph.

Enables Observability: Tracing is only trustworthy if it’s stitched to the right identity with lineage.

Backs the CDP App: Governed real-time profile views for activation on the event backbone.

How Some Banks Handle Fulfillment and Decisioning… and Why It Breaks at Scale

Many institutions still operate fulfillment and decisioning via a patchwork of scheduled/ad-hoc SQL scripts tied to specific offer variants and a marketing data mart whose inputs lag by days—forcing exceptions and direct taps into operational systems for “fresh” data.

Resulting Issues Seen in the Field

Fragile Codebases:
Every new offer = new script; reuse bloats complexity.

Stale Decisions:
EDW/mart refresh cycles miss real-time behaviors.

Role Mismatch:
Analysts maintain operational SQL; they're not staffed to engineer at scale or securely.

Financial Logic is Specialized:
Rolling balances, new-money detection, tiered rewards, promo code lifecycle, exceptions, regulatory controls.

Shadow IT Risk:
Unvetted pipelines, no monitoring/failover, weak change management; elevated security, availability, continuity, and compliance risks.

What a Composable Customer Data Platform Changes

A governed event backbone, real-time decisioning, and a shared offer/profile repository replace one-off scripts with declarative logic, observable dataflows, and environment-promotable configurations—so marketing moves fast without breaking IT standards.

Why a Composable CDP Works

Speed and Relevance

Millisecond decisions by combining streaming detection with immediate historical lookups.

Consistency

A unified object graph keeps decisioning, delivery, and fulfillment aligned.

Agility

Declarative rules and flexible schemas enable rapid iteration without heavy refactoring.

Business Value for IT and the Enterprise

Reduced Complexity and Cost:
Fewer integration points and lower maintenance overhead than multi-stack approaches.

Data Governance Simplified:
One clear flow eases security and privacy enforcement.

Faster Time-to-Market:
Modular updates and parallel workstreams across teams.

End-to-End Traceability and Analytics:
Full journey tracing and actionable insights for optimization and compliance.

Scalability at Financial Scale: Billions of Records, Millions of Events per Hour

Horizontally Scalable Services:
Stateless decision workers and sharded context stores scale linearly; back-pressure and idempotent processing protect downstream systems.

Keyed Partitioning:
Customer/account-key partitioning keeps hot paths fast and predictable even as data volumes grow.

Zero-Copy Data Access:
Reading from data platforms like Snowflake via secure sharing avoids ETL sprawl and storage duplication while letting you scale on warehouse elasticity.

Unified Offer Repository:
One canonical offer record eliminates cross-store joins at runtime, cutting latency and error rates at high throughput.

Data Consistency

Dimension	Traditional Scripts & Data Mart	Composable Event Platform
Freshness	Daily/weekly; often stale	Streaming + sub-second context lookups
Logic	Per-offer SQL; brittle reuse	Declarative, reusable rules; modular updates
Governance	Ad-hoc exceptions; shadow IT	One flow with enforced policies and audit trails
Observability	Limited	End-to-end tracing and actionable insights
Data consistency	Many stores, drift risk	Unified object graph with governed repositories for offers/profiles

Architecture Benefits Summary

The architecture delivers what financial institutions need:

A platform fine-tuned for banking that bakes in domain semantics and relationship-aware identity, so outcomes are correct by construction; transactional integrity and compliance with per-customer/account ordering, exactly-once processing, rollback/replay, and policy-at-read for audit-ready decisions and fulfillment.

A composable, warehouse-native CDP that runs next to governed data via secure sharing/peering (zero new copies/engines), pushes down heavy work, and uses a hot context tier for sub-second lookups.

A unified object graph with lineage—separate, versioned objects (Offer, Product, Disclosure, Content, Journey, Fulfillment) linked by explicit relationships—so changes propagate safely, and decisions are explainable.

The result is a single architecture that powers multiple apps (Fulfillment, Journeys, CDP activation, Offer Decisioning) for reuse over rebuild, with millisecond-level responsiveness, historical context, end-to-end traceability, and stateless horizontal scale to billions of records/events, all while reducing integration surface area and ongoing maintenance.

Implementation Notes

Fine-Tuned for Financial Services Use Cases

Canonical Objects and Events:
Define schemas for Offer, Product, Disclosure, Content Block, Journey Step, Fulfillment Event; events for Customer, Account, Interaction, Fulfillment.

Domain Semantics:
Encode rolling balances, new-money detection, tiered incentives, promo lifecycle, jurisdictional disclosures, and household/role logic.

Deterministic Disclosures:
Bind legal language from rules tied to product/offer attributes—not free-text generation.

Operational Playbooks:
Runbooks for incentive posting, exception handling, and remediation; KPIs for acceptance, fulfillment exceptions, audit turnaround.

Transactional Integrity and Compliance

Ordering and Idempotency:
Enforce per-key (customer/account) ordering on streams; stamp idempotency keys on acceptance/fulfillment; implement exactly-once writes to downstream systems.

Rollback and Replay:
Maintain deterministic replay windows with lineage when an upstream feed is corrected.

Policy at Access:
Apply consent, purpose limits, and entitlements **at read** in profile/decision APIs; minimize attributes by purpose; mask/tokenize where not strictly needed.

Reconciliation:
Compare expected vs. actual postings/balances via **read-only** warehouse/lake checks; route mismatches to a managed exception queue.

Data Model (Unified Object Graph)

Version and Relate:
Store each object separately with versions and explicit relationships; never “stuff” a mega-record.

Identity and RCP:
Resolve party/account/household at access time; preserve history across temp-to-perm IDs and M&A.

Lineage and Impact:
Record which inputs/policies produced each outcome; surface downstream impacts before changes go live.

Composable CDP (Warehouse-Native)

Zero-Copy Reads:
Integrate via secure sharing/peering; use governed tables/views as inputs.

Push-Down and Hot Context:
Delegate heavy joins to the warehouse/lake; serve sub-second lookups from a context tier with cache invalidation by event/TTL.

Operational Write-Backs Only:
Persist just the operational objects needed for activation/audit (e.g. offer, fulfillment status).

SLOs and Fallbacks:
Define p95 latency targets for decisions and warehouse reads; implement graceful degradation paths.

What to Remember

Fine-Tuned for Financial Services:
Bank-grade semantics (balances, “new-to-bank,” tiered rewards, disclosure obligations) and relationship-aware identity/RCP ensure what you design can be fulfilled and defended.

Transactional Integrity and Compliance:
Per-customer/account ordering, exactly-once processing, rollback/replay, and policy-at-read make outcomes correct by construction and audit-ready.

Relationship-Aware Data Model:
Separate, versioned objects —Offer, Product, Disclosure, Content Block, Journey Step, Fulfillment Event, Party/Account/Household— linked with lineage and impact analysis, so changes propagate safely, and decisions are explainable.

Composable CDP (warehouse-native):
Bring decisions to governed data via secure sharing/peering (zero new copies/engines); push down heavy work, keep hot context for sub-second lookups, and materialize only operational records

Scale By Design:
Stateless workers, per-key partitioning, and zero-copy access support billions of records/events without brittle pipelines—so you can ***design → decide → deliver → accept → fulfill → reconcile → learn*** on one architecture.

Conclusion

From data platform to execution layer.

A real-time, composable CDP for financial services must meet four requirements:

1. Be fine-tuned to banking use cases
2. Guarantee transactional integrity and compliance
3. Operate on a relationship-aware data model
4. Run warehouse-native with zero-copy access.

Institutions that adopt these pillars can design, decide, deliver, accept, fulfill, reconcile, and learn on one architecture—turning customer intent into auditable, repeatable outcomes and shortening the path from AI pilots to production. Start with a closed-loop fulfillment slice; reuse the same capabilities for journeys and decisioning. The result isn’t another data copy—it’s a CDP that the business can trust.

