WHITE PAPER

Transforming Legacy Appian Environments:

A Roadmap to Enterprise Data Fabric

By Brian Kim, REI Solution Architect and
Chetan Rane, REI Offering Lead for Digital Transformation

REI SYSTEMS

2026

CONTENT

03 ***Introduction***
The Challenge

04 ***The Limits of Legacy:*** *Analyzing Traditional Data Access Patterns*

05 ***Modern Data Fabric Architecture:*** *Enabling Unified, Real-Time Enterprise Operations*

06 ***The Enterprise Integration Gap:*** *Designing for Scale and Complexity*

07 ***The Data Virtualization Abstraction Layer***

08 ***Architectural Patterns:*** *Thick vs. Thin Logical Data Fabric*

09 ***Architectural Judgement:*** *When Data Fabric Alone is Not Enough*

10 ***Roadmap to Transformation:*** *A Phased Migration Strategy*

12 ***Technical Deep Dive:*** *From CDT to Data Fabric Components*

13 ***Governance, Security, and AI-Readiness***

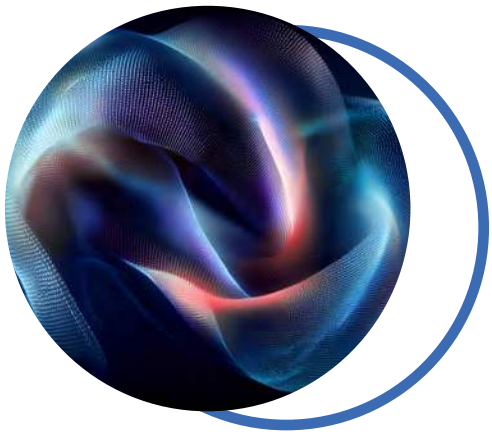
14 ***Conclusion:*** *Strengthening the Enterprise Data Foundation*

15 ***About the Authors***
Appendix: *Acronyms*

Introduction

Enterprises are under increasing pressure to modernize legacy application environments while supporting real-time operations, high-volume transactional processing, and emerging Generative AI (GenAI) capabilities. Many organizations running older Appian implementations rely on high-latency data access patterns, including direct relational database management system (RDBMS) queries, tightly coupled Custom Data Types (CDTs), Data Store Entities (DSEs), and unsynced Record Types. While once sufficient, these approaches now introduce architectural constraints that limit scalability, increase the Total Cost of Ownership (TCO), and restrict the agility required for sustained digital transformation. These limitations directly affect an organization's ability to deliver sophisticated reporting, large-scale case management, and cross-system orchestration at enterprise scale.

Modern enterprises require a unified, virtualized data architecture that supports transactional and analytical workloads without costly migration. Gartner reports that implementing a data fabric can reduce integration design time by 30 percent, deployment time by 30 percent, and maintenance time by up to 70 percent by shifting effort from fragile custom integrations to semantic modeling and composable architecture [1]. Strengthening the data foundation is essential for performance, centralized governance, and AI readiness.



The Challenge

Legacy Appian architectures rely on tightly coupled data access models that introduce operational friction and financial risk. Heavy use of queryEntity() against external RDBMS systems creates latency and performance variability, particularly with cross-system joins and network dependencies. CDT and DSE mappings are directly tied to physical database schemas, requiring manual remediation when schemas evolve and slowing release cycles. Unsynced record types limit centralized, real-time composite views needed for enterprise reporting and large-scale case management. As data volumes grow, manual database tuning by DBAs becomes necessary to sustain performance, increasing TCO and intensifying scalability pressures as synchronization approaches limits of up to 20 million rows per record type.

Governance gaps compound these constraints. Row-level access controls enforced at the interface layer rather than centrally create inconsistent security and heightened compliance exposure, especially in FedRAMP High and IL5 DISA PA environments. Without a unified, governed semantic data layer, organizations lack the architectural foundation required for Retrieval Augmented Generation (RAG), enterprise semantic search, and secure Generative AI adoption. Data remains fragmented across siloed systems, limiting accurate, context-aware AI outputs and making modernization essential for enterprise-grade scalability, governance, and composable application design.

The Limits of Legacy: Analyzing Traditional Data Access Patterns

A formal architectural critique of pre-data fabric methods reveals inherent limitations that make modernization mandatory for enterprises seeking scalable, secure, and agile operations.

Technical Friction Points in Pre-Data Fabric Architecture

The reliance on Custom Data Types (CDTs) and Data Store Entities (DSEs) creates brittle system architecture. When changes occur in the underlying RDBMS schema, developers are forced into manual, time-consuming, and often fragile updates to the corresponding CDT and DSE mappings. This tight coupling between the application layer and the physical database structure is fundamentally incompatible with the demands of modern agile development, which requires frequent releases and schema flexibility.

Similarly, performance bottlenecks are common in legacy systems that rely heavily on direct queryEntity() calls. These queries frequently lack optimization and rely heavily on efficient network connections to the source RDBMS, often leading to unacceptable latency. The complexity of managing Create, Read, Update, and Delete (CRUD) operations across multiple external systems requires custom, bespoke logic, substantially increasing development complexity and the persistent risk of data inconsistencies. Older, unsynced record types offer limited scalability and fail to provide the centralized, real-time data view essential for sophisticated reporting, case management, and orchestration.

Governance, Security, and Scalability Deficiencies

In legacy architectures, security enforcement, specifically access control for sensitive rows, is fragmented and must be implemented repetitively at the application interface level. This complexity creates increased security exposure and noncompliance risk, as security policies are defined in disparate locations rather than governed centrally.

Operational costs are also inflated due to the lack of built-in performance optimization. Pre-data fabric scalability is heavily dependent on expensive, dedicated database administrators (DBAs) performing manual performance tuning and index optimization. The persistence of this manual labor is a high-cost operational burden that data fabric's auto-optimization capability is specifically engineered to eliminate. This high TCO caused by perpetual manual tuning is a direct financial justification for migrating to a modern, automated data platform.

Finally, while data warehouses and data lakes are well suited for historical, analytical, and immutable data, they inherently struggle with mutable, real-time transactional data. Older Appian architectures often inherit this problem, making it exceptionally difficult to build truly responsive, high-volume transactional applications, such as core financial or case management systems. These limitations of CDTs and direct queries represent the technical barriers preventing older Appian applications from achieving the enterprise-grade status now mandated for modern low-code application platforms (LCAPs).

Modern Data Fabric Architecture: Enabling Unified, Real-Time Enterprise Operations

A modern Data Fabric architecture addresses the scalability, governance, and integration limitations inherent in legacy application environments. Rather than relying on tightly coupled database integrations and custom synchronization logic, a Data Fabric introduces a unified, virtualized semantic layer that connects disparate systems without requiring large-scale data migration.

Architectural Pillars: Virtualization, Unification, and Real-Time Access

At its core, a Data Fabric establishes a unified data layer that abstracts the physical location and structure of enterprise data across on-premises RDBMS environments, ERP systems, CRM platforms, and cloud services. By virtualizing access rather than extracting and duplicating data, organizations reduce migration risk, shorten implementation timelines, and preserve the integrity of source systems.

This architecture supports both mutable transactional data and immutable analytical data within a governed model. As a result, organizations can operate effectively in real-time environments such as complex case management, financial processing, and partner data-sharing ecosystems. Real-time Create, Read, Update, and Delete (CRUD) actions can be executed directly against source systems while maintaining centralized access control.

When implemented using modern low-code platforms, including Appian where appropriate, this model leverages record-based constructs and semantic relationships to unify data views without duplicating underlying systems of record. It enables scalable operational workloads while maintaining practical synchronization thresholds, including volumes of up to 20 million rows per record type.

Enterprise Scale and Architectural Considerations

Enterprise Data Fabric implementations must support high-volume datasets, performance optimization, and centralized governance without introducing excessive operational overhead. Modern platforms incorporate automated performance optimization mechanisms that adjust based on usage patterns, reducing reliance on manual database tuning, and lowering long-term TCO.

Data lakes and data warehouses primarily support historical analytics and typically require extraction and replication of source data. Data mesh approaches distribute ownership across services and rely heavily on API orchestration and custom integrations. A Data Fabric architecture instead emphasizes virtualization and semantic modeling, providing unified access across distributed systems without large-scale duplication or extensive custom code.

For REI Systems, the focus is not on promoting a single technology, but on designing architectures that align platform capabilities to mission requirements. Within this approach, a well-implemented data fabric strengthens governance, improves scalability, reduces integration complexity, and establishes the data foundation required for advanced analytics, enterprise reporting, and secure GenAI adoption.

The following table summarizes the fundamental transformation resulting from the architectural shift.

Feature	Legacy (CDTs/direct query)	Appian Data Fabric (modern records)	Architectural benefit
Data location	Fragmented; requires complex ETL/migration	Virtualized layer (connects sources)	Eliminates migration debt; 30-70% faster integration
Data model	Code-dependent, fragile mappings (CDT)	Codeless, abstract, semantic layer	Democratizes data modeling; reduced time-to-value
Data mutability	High latency, complex manual write-back	Real-time CRUD (Read/ Write) to source	Supports real-time transactional workflows (e.g., case management)
Scalability	Manual DBA tuning required; often volume-limited	Auto-optimized, high-volume sync (up to 20M rows)	Enterprise-grade performance; significant TCO reduction
Security	Custom-implemented at the application layer	Native row-level security (RLS) enforcement	Centralized governance and compliance confidence

Table 1. Comparison of Data Access Approaches

Centralized Governance and Compliance

A Data Fabric delivers a centralized, comprehensive approach to data security. Row-Level Security (RLS) rules are enforced directly on the composite data model, ensuring centralized control over who can view, update, or delete specific data subsets across all connected systems. This centralized data management approach is essential for maintaining the rigorous security and compliance required for highly sensitive environments, including US government clients operating under FedRAMP High and IL5 DISA PA authorizations.

The Enterprise Integration Gap: Designing for Scale and Complexity

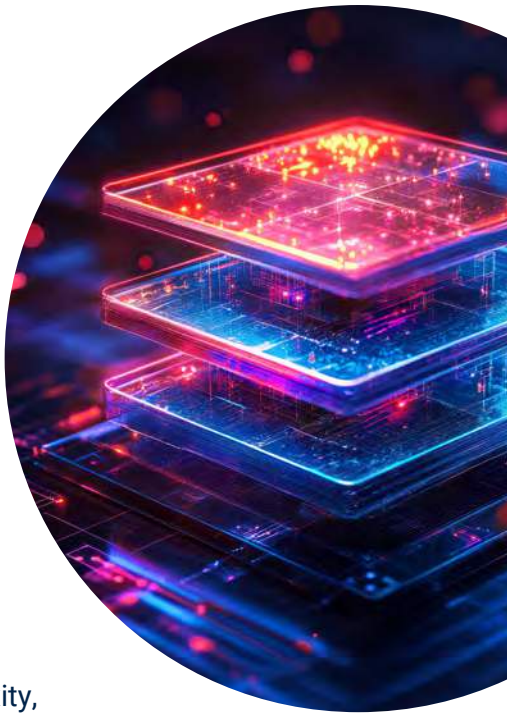
While modern low-code platforms provide native data virtualization and synchronization capabilities, large-scale legacy modernization efforts often introduce integration challenges that require additional architectural consideration. In complex enterprise environments, relying solely on native synchronization features may introduce design risk when data scale, source complexity, or transactional sensitivity exceeds platform thresholds.

In REI's experience supporting federal modernization initiatives, these risks typically emerge in four scenarios.

1. **Massive Data Volumes:** Synced Record Types support substantial operational datasets, including volumes up to 20 million rows per Record Type. However, enterprise systems may contain tens or hundreds of millions of historical and transactional records, where full synchronization can introduce performance constraints or exceed practical limits.
2. **Evolving Platform Economics and the Sync Limit Decision:** Appian is moving toward offering unlimited row synchronization, which on the surface resolves the volume constraint. However, this

expanded capacity comes at additional cost, fundamentally changing the nature of the decision. What was previously a technical problem to architect around becomes a budget decision to justify. For clients operating under fixed-price contracts or constrained modernization budgets, the cost of unlimited sync must be weighed honestly against alternative architectural patterns such as data virtualization or database segmentation. The right answer is no longer purely technical, and that distinction matters early in the engagement.

3. **Complex Legacy Integration:** Legacy databases frequently include segmented schemas, non-standard data types, database-side triggers, proprietary APIs, and extensive stored procedures. Mapping these structures directly into simplified application-level record models can create fragility, limit maintainability, and increase long-term risk.
4. **Database Burden:** Direct synchronization or integration against high-volume Online Transaction Processing (OLTP) systems can introduce unacceptable load on mission-critical source databases, particularly in environments requiring continuous uptime and strict service-level performance guarantees.



Addressing these constraints does not require abandoning a Data Fabric strategy. Instead, it requires architectural layering. In large-scale environments, REI designs a federated integration model that introduces a dedicated data virtualization layer between application platforms and complex legacy systems. This layer abstracts source complexity, optimizes federated query execution, and protects core systems from excessive load while preserving a unified semantic model for application consumption.

Data virtualization platforms, such as Denodo when appropriate, can encapsulate stored procedures, complex joins, and heterogeneous data structures, exposing standardized, query-optimized views to application layers. This approach allows the application platform to operate within practical synchronization thresholds while maintaining enterprise-scale access to distributed data.

REI's value lies in determining when native platform capabilities are sufficient and when an additional semantic abstraction layer is required. By designing integration architectures aligned to data volume, system complexity, compliance requirements, and performance expectations, REI ensures that modernization efforts remain scalable, governed, and sustainable over time.

The Data Virtualization Abstraction Layer

Data Virtualization (DV) is positioned between the Appian platform and the legacy data sources, solving these problems by effectively hiding the complexity.

1. **Hiding Massive Volumes:** A platform like Denodo does not move data; instead, it uses advanced techniques - like query pushdown, intelligent caching, and parallel processing - to query only the necessary data segments directly at the source. This allows Appian to access billions of records without hitting sync limits or requiring Appian to process the entire dataset. DV aggregates, filters, and joins data closer to the source, then delivers only the small, relevant, and query-optimized result set to Appian.

2. **Harmonizing Legacy Complexity:** DV tools can encapsulate stored procedures, complex joins across federated databases, and proprietary data structures (like mainframes or NoSQL) and expose them to Appian as a single, clean, ANSI-SQL-compliant virtual view. This bypasses Appian's constraints on direct stored procedure calls or complex schema mapping, delivering a simplified, business-friendly data model ready for the Appian environment.
3. **Optimizing Query Performance:** The virtualization layer Cost-Based Optimizer (CBO) determines the most efficient way to execute a federated query across multiple, distributed sources. This ensures Appian applications maintain high performance without requiring manual index tuning or suffering slowdowns associated with cross-source joins.

Architectural Patterns: Thick vs. Thin Logical Data Fabric

By utilizing a dedicated DV layer, the roadmap to modern Appian architecture offers two primary logical data fabric patterns:

Thick Data Fabric Pattern (Recommended for Scale and Complexity)

This pattern leverages a dedicated data virtualization layer to handle the most challenging integration scenarios.

Component	Function	Advantage
Appian Data fabric	**Consumer layer:** Provides visual record types, relationships, and low-code data modeling for end-user applications and Process HQ.	**Agility and UX:** Fast development, native security, and AI assistant capabilities.
Data virtualization (Denodo)	**Semantic layer:** Handles complex joins, data quality, security enforcement, and source abstraction across all underlying systems.	**Performance and scale:** Bypasses Appian sync limits, optimizes complex queries, and integrates legacy logic (stored procedures, triggers).
Legacy systems	**Source layer:** The physical data remains in place in the ERP, CRM, data warehouse, or cloud storage.	**Integrity:** Protects core OLTP systems from excessive application query load.

Table 2. Thick Data Fabric Pattern

Thin Data Fabric Pattern (Recommended for Simplicity and Targeted Use Cases)

This pattern relies primarily on Appian's native integration capabilities, with the DV layer being minimal or absent, leveraging the Data Fabric's built-in features for data access.

Component	Function	Advantage
Appian data fabric	**Consumer & integration layer:** directly manages data via synced record types (within limits) or unsynced record types using **web services/ low-code integrations.**	**Cost-efficiency & simplicity:** minimal infrastructure overhead; leverages Appian's native sync and security features directly.

Minimal federation	**Abstraction layer:** handled by Appian's native tools (e.g., data store entities, integration objects).	**Speed for simple needs:** quick connection and access for small-to-midsize, less complex data sources.
Legacy systems	**Source layer:** directly exposed to Appian via connectors.	**Direct access:** good for read-only or small transactional updates where latency is tolerable.

Table 3. Thin Data Fabric Pattern

This hybrid approach allows clients to leverage the phenomenal speed of Appian's low-code development while ensuring the enterprise data foundation is robust, scalable, and capable of handling even the most challenging legacy environments. It transforms the limitations of Appian's native sync into a strategic integration point for best-of-breed virtualization technology when a Thick Fabric is required.

Architectural Judgement: When Data Fabric Alone is Not Enough

Real-world federal modernization rarely presents clean architectural choices. While Appian's synced Record Types deliver significant value as the foundation of a Data Fabric strategy, REI's delivery experience across complex government environments has revealed that prescriptive, one-size-fits-all approaches consistently underperform. The right architecture is always determined by client constraints, not platform capabilities alone.

When a Data Virtualization Middle Tier Makes Sense

In environments where legacy data complexity or volume exceeds practical sync thresholds, introducing a data virtualization layer between the source system and Appian can be highly effective. By exposing a clean, optimized virtual view to Appian, synced Record Types can still be created and maintained, preserving all the benefits of Data Fabric while abstracting the underlying complexity. In this pattern, Data Fabric remains intact. The virtualization layer simply makes it viable at enterprise scale.

However, this approach is not universally appropriate. Many federal clients operate under constraints that make an intermediary layer impractical or prohibited. Procurement and licensing restrictions may prevent the introduction of additional third-party platforms altogether. Security and authorization boundaries can limit what systems are permitted to sit between Appian and a sensitive or classified data source. Architectural governance policies sometimes mandate direct system-to-system integration patterns with no exceptions. In other cases, cost constraints make a best-of-breed data virtualization platform financially unjustifiable given the scope of the effort.

When Database Segmentation Drives a Different Pattern Entirely

One of the more complex scenarios encountered in federal delivery involves clients who, due to data scale and performance requirements, have deliberately segmented their databases and tables by case or record type. In these environments, a particular case record may resolve to one database instance while a different case type resolves to an entirely separate database, each with its own schema, access controls, and performance profile.

In this scenario, a single synced Record Type strategy breaks down. There is no clean unified view to sync against, no virtualization layer that satisfies all constraints, and no single architectural pattern that fits every workload. The solution requires a hybrid approach that may combine synced Record Types where feasible, direct integration patterns where necessary, and careful orchestration logic to route data access appropriately, all while maintaining the governance and security posture required in federal environments.

What This Means in Practice

These scenarios reinforce a consistent reality: technology is not the limiting factor. Successful modernization depends on aligning architecture to client environment, data scale, procurement constraints, security posture, and operational requirements.

The value in these engagements is not familiarity with Appian's feature set. That knowledge is expected. The differentiator is the ability to apply delivery judgment to select the appropriate architectural pattern, determine when Data Fabric is sufficient, and identify when it must be augmented or replaced to meet mission needs.

This judgment is grounded in experience designing scalable data and application architectures across complex environments. The principles of virtualization, federation, semantic abstraction, and governed data access are well established. What has evolved is the maturity of the tooling that enables these patterns to be implemented effectively at enterprise scale.

Applying these patterns based on context is what distinguishes an experienced architecture partner from one that is simply following a vendor roadmap.

Roadmap to Transformation: A Phased Migration Strategy

Migrating large, legacy Appian applications to the Data Fabric architecture requires a structured, four-phase roadmap designed to mitigate risk and ensure alignment with strategic business goals.

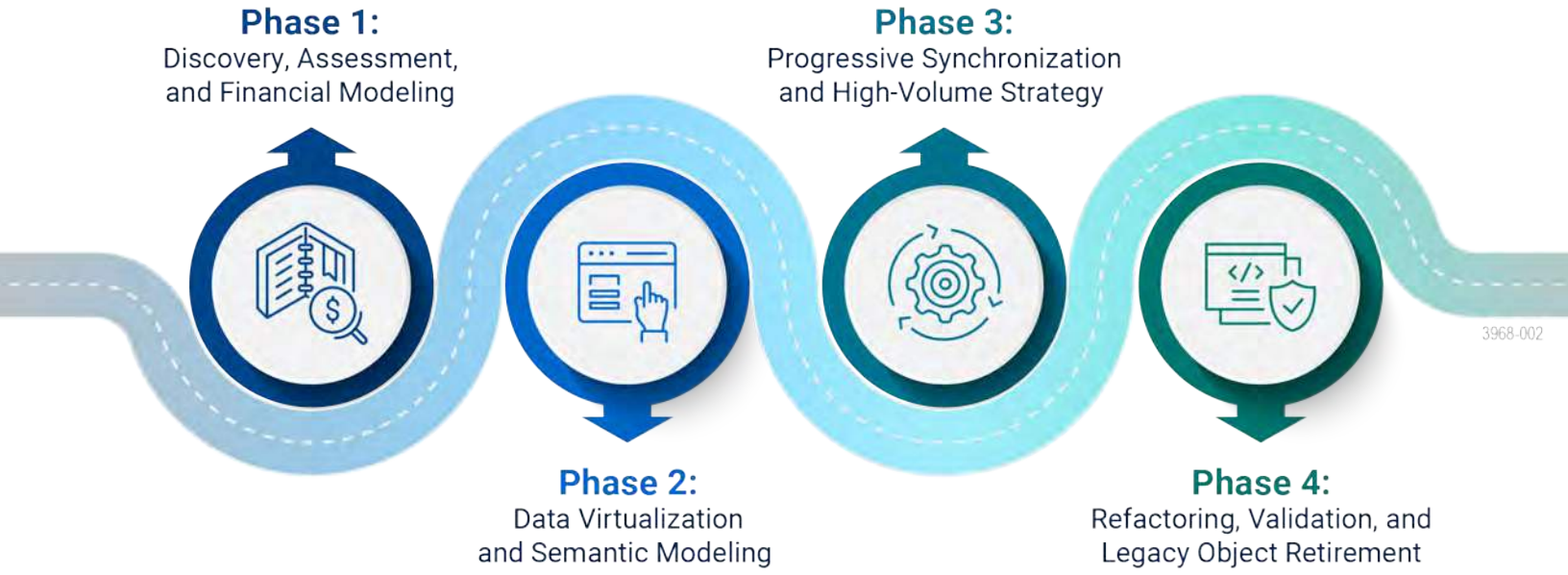


Figure 1. Roadmap to Transformation

Phase 1: Discovery, Assessment, and Financial Modeling

The process begins with a comprehensive audit of all existing data objects, including CDTs, DSEs, complex query rules, and application dependencies, alongside identification of all external data sources (RDBMS, web services). Core business entities must be mapped from the legacy structures. Prioritization should be based on business criticality and the severity of existing performance or security pain points.

A critical financial assessment must be conducted early in this phase. Technology Business Management (TBM) and Financial Operations (FinOps) principles require clear ownership of usage and a defined understanding of current-state costs. Before migrating any application, organizations should establish a fully loaded TCO baseline. This baseline enables ROI validation post-migration and supports informed, defensible decision-making throughout the process.

As organizations transition to a Data Fabric architecture, it becomes essential to tag, catalog, and allocate workflows to accurately attribute costs across systems, applications, and functional owners. Particular attention should be given to consumption-based costs, ensuring each application owner understands key cost drivers and their associated operational impact.

Additionally, anecdotal evidence suggests that a significant failure point in enterprise migrations occurs when clients fail to upgrade their platform licensing. Many new, advanced Data Fabric capabilities, including high-volume synchronization and enterprise-grade AI integration features, are often only available at the Advanced or Premium pricing tiers.1 To avoid functional gaps or performance ceilings post-migration, the budget must proactively include this necessary platform uplift.

Phase 2: Data Virtualization and Semantic Modeling

The developer team must define Appian Record Types as the singular source of truth for each business entity. This involves designing the semantic layer, which uses a codeless data model to abstract the underlying complexity of disparate source systems. The connection strategy must configure Record Type sources (External Database, Web Service) and accurately define one-to-many and many-to-one relationships, replacing the need for complex, manual join logic in application code. Critically, the semantic model must be structured to unify case data across related record types (e.g., Accounts, Customers, Regions) to ensure robust preparation for subsequent process insights.

Phase 3: Progressive Synchronization and High-Volume Strategy

This phase initiates data movement and governance activation. Synchronization should target the most active, transactional, and medium-volume datasets where real-time CRUD access is essential.

For high-volume datasets that approach or exceed the 20 million row threshold, a hybrid architectural strategy is required to mitigate performance failures. Enterprises must use the “Keep data available at high volumes” sync option to dynamically synchronize only the latest, most active rows of data, effectively preventing the record type from breaching synchronization limits. Historical, largely static data must be left virtualized (unsynced) to maintain the overall data fabric view while optimizing performance for the active operational window. This deliberate segmentation prevents the failure of treating the Data Fabric as a historical data warehouse, which violates the architectural intent of the platform.

Simultaneously, all Row-Level Security (RLS) rules defined during Phase 2 must be activated and rigorously tested. Activating governance early ensures that security controls are integrated into the new data access layer from the outset, accelerating user adoption of the unified views and avoiding potential data exposure during the refactoring process.

Phase 4: Refactoring, Validation, and Legacy Object Retirement

Developers must systematically refactor application logic by replacing all references to legacy objects. This involves substituting all deprecated DSE functionality and older queryEntity() calls with the modern, optimized queryRecordType() functions and record-powered interfaces. Rigorous performance testing is mandatory to ensure the new Data Fabric queries meet or exceed existing application Service Level Agreements (SLAs). Finally, legacy CDTs, DSE configurations, and underlying custom synchronization code should be formally archived and retired, completing the elimination of architectural debt.

Migration Checklist: From CDT/DSE to Data Fabric

Roadmap phase	Legacy action/ pain point	Data Fabric action/solution	Key risk mitigation strategy
Discovery/ phase 1	Under-budgeting for enterprise scale/security features	Budget for advanced/ premium tiers required for RLS and high volume sync	Verify feature requirements against Appian pricing tiers proactively
Modeling/ phase 2	Brittle CDT-to-DB mapping and synchronization logic	Define record types, relationships, and field visibility	Rigorously test conceptual model against complex legacy relationships
Synchronization/ phase 3	High latency, complex manual write-back	Real-time CRUD (Read/ Write) to source	Supports real-time transactional workflows (e.g., case management)
Refactoring/ phase 4	Manual DBA tuning required; often volume-limited	Auto-optimized, high-volume sync (up to 20M rows)	Enterprise-grade performance; significant TCO reduction

Table 4. Migration Checklist

Technical Deep Dive: From CDT to Data Fabric Components

Successfully executing the migration requires developers to adopt modern data engineering principles, moving away from RDBMS-centric logic toward semantic modeling.

Schema Evolution and Drift Management

Developers must efficiently translate complex or deeply nested CDT structures into optimized, properly related Record Types. A core technical challenge of data virtualization is managing schema drift, unanticipated changes in the external source systems. The strategy to handle this must incorporate disciplined Data Ops practices. This includes running automated profiling jobs to inspect and observe field-level changes in external schemas over time. Crucially, strict schemas should be enforced only

at the key logical boundary of the Record Type layer, providing tolerance upstream. Finally, datasets or records must be tagged with an explicit schema version to ensure traceability and backwards compatibility. Integrating compatibility testing within the continuous integration/continuous deployment (CI/CD) pipeline is mandatory before promoting schema changes to production. The necessity of actively managing schema drift requires developers to apply version control and testing rigor to their semantic models, ensuring that the promised maintenance reduction is achieved through structured development.

Data Events, Transactions, and Auditing

The Data Fabric streamlines transaction handling and auditing. Developers should utilize the platform's built-in configuration of Record Events to automate data. This method provides the simplest and most robust way to generate a clean, structured audit trail and event history, which is necessary for process insights, effectively replacing complex, legacy logging mechanisms. Additionally, developers must configure Record Actions (Write) to ensure that real-time data updates flow seamlessly and securely back to the source ERP or CRM systems, confirming the platform's real-time transactional capability.

CI/CD Integration and Metadata Management

The Data Fabric data model, comprising Record Types, relationships, and RLS rules, is configured via the Appian Designer front end. This means that the system metadata defining this semantic layer must be formally treated as deployable artifacts within the CI/CD pipeline. Developers must align DevOps practices to integrate these Data Fabric objects into external tooling via Appian's APIs, ensuring consistent and governed promotion of the semantic layer across development, testing, and production environments. This process supports the LCAP requirement for extensibility and provides the necessary structure to guarantee the architectural stability of the unified data layer.

Governance, Security, and AI-Readiness

The Data Fabric migration provides long-term strategic value by serving as the foundation for governed data democratization and the enterprise's future GenAI strategy.

Centralized Security and Compliance Foundation

Row-Level Security (RLS) is enforced across the composite data view, allowing organizations to expand secure data access to line-of-business employees and partners (data democratization) with confidence. IT teams gain a centralized view of all security controls, simplifying compliance reporting and supporting the stringent requirements necessary for mission-critical deployments, such as FedRAMP High and IL5-authorized GovCloud environments. This centralized security model acts as an accelerator, as new composite data views and applications can be spun up faster without requiring extensive, redundant security reviews for every new service.

Data Democratization and Actionable Insights

By presenting a simplified, unified data model through Record Types, the platform empowers non-developer users to explore data, build reports, and gain actionable insights in real time. The integration of the Data Fabric with Key Performance Indicators (KPIs) for business analytics ensures that operational and process performance metrics are unified, contributing to improved overall platform operations and better business outcomes.

Data Fabric as the Pre-requisite for Generative AI (GenAI)

Migration to a Data Fabric architecture is foundational for future AI capabilities. By extending governance beyond structured tables to include documents through a "Document-as-Record" model, unstructured content becomes part of the enterprise semantic layer rather than remaining isolated files. This enables secure semantic search and provides the structured context required for AI systems to retrieve and ground responses in authoritative data sources

Crucially, this unified, structured data access provides the critical architectural step required for building Retrieval Augmented Generation (RAG) pipelines. The Data Fabric serves as the secure, governed data source (grounding) for large language models (LLMs). This grounding ensures that AI agents provide accurate, context-aware, and secure results, directly supporting the LCAP market trend toward embedded AI assistant and Agentic AI capabilities. Without the Data Fabric unifying and governing access to both structured and unstructured information, the implementation of secure, enterprise-scale GenAI is severely limited.

Conclusion: Strengthening the Enterprise Data Foundation

Migrating from legacy data access patterns—including CDTs, unsynced records, and direct RDBMS queries—to a modern Data Fabric architecture is not simply a platform enhancement; it is a structural modernization of the enterprise data foundation. This shift reduces architectural debt, improves long-term maintainability, and enables measurable efficiency gains, including maintenance reductions of up to 70 percent as reported in data fabric implementations. It also supports scalable operational workloads, including synchronization thresholds of up to 20 million rows per Record Type when architected appropriately.

More importantly, a governed, unified semantic data layer establishes the foundation required for enterprise reporting, real-time case management, cross-system orchestration, and advanced analytics. It enables secure adoption of Gen AI, RAG, and process mining capabilities by grounding AI systems in structured, authoritative data sources under centralized access control.

For REI Systems, modernization is not about platform promotion. It is about designing integration architectures that align data scale, system complexity, compliance requirements, and mission objectives. Organizations that strengthen their data architecture today position themselves for sustained performance, controlled innovation, and long-term operational resilience.

About the Authors

Brian Kim

REI Solutions Architect

Brian Kim is an accomplished technology leader with 25+ years of experience defining and executing digital transformation including data strategies for government organizations, including Freddie Mac, DoD, DHS, US Army, and NASA HQ.

Contact him at: brian.kim@reisystems.com

LinkedIn

Chetan Rane

REI Offering Lead for Digital Transformation

Chetan Rane is the Offerings Lead for Digital Transformation at REI Systems. He has over a decade of experience driving innovative solutions in government IT modernization and digital transformation.

Contact him at: crane@reisystems.com

LinkedIn

Appendix: Acronyms

Acronym	Definition
CDT	Custom Data Type
CI/CD	Continuous Integration / Continuous Deployment
CBO	Cost-Based Optimizer
CRUD	Create, Read, Update, Delete
DISA	Defense Information Systems Agency
DSE	Data Store Entity
DV	Data Virtualization
ERP	Enterprise Resource Planning
ETL	Extract, Transform, Load
FedRAMP	Federal Risk and Authorization Management Program
GenAI	Generative Artificial Intelligence

Acronym	Definition
IL5	Impact Level 5 (DoD Cloud Security Classification)
KPI	Key Performance Indicator
LCAP	Low-Code Application Platform
LLM	Large Language Model
OLTP	Online Transaction Processing
RAG	Retrieval Augmented Generation
RDBMS	Relational Database Management System
RLS	Row-Level Security
SAP	Systems, Applications, and Products (ERP platform)
TCO	Total Cost of Ownership

Footnotes

1. Gartner, "Data Fabric," https://www.gartner.com/en/data-analytics/topics/data-fabric

Copyright © 2026 REI Systems. All rights reserved.