Transaction Inventory

Jim Episale

ABSTRACT

In complex enterprise ecosystems, managing transactional workflows across multiple platforms poses challenges in integration, traceability, and system scalability. Traditional application-focused integration approaches lead to excessive API proliferation, technical debt, and inefficient workflow management. This paper introduces the **Transaction Inventory Model**, a paradigm that treats transactional data like product inventory in a supply chain—ensuring identification, categorization, and streamlined distribution across an enterprise architecture.

By leveraging standardized metadata structures, barcoding principles, and event-driven middleware, this model provides a configurable, application-agnostic integration framework. It simplifies data transformation, enhances visibility, and centralizes workflow management, reducing redundant customizations. Furthermore, an event-driven architecture minimizes direct API dependencies, fostering loose coupling and asynchronous processing.

The proposed approach not only improves efficiency but also ensures adaptability to evolving enterprise technologies without costly system overhauls. By applying inventory control and distribution logistics principles, organizations can achieve a scalable, resilient, and future-proof integration ecosystem.

1. CONTEXT

Never implement or create something that has only one use. A principle learned as a sergeant for a reconnaissance platoon that you always fill your pack with essential equipment that has more than one use due to the length of which we had to carry. This concept drives simplification of integration architectures that will be demonstrated in this article.

Early in my career, I spent time as a data integration developer for a food distribution company. Because the margins are so low in that business everything you do has to be simple, low cost and reusable. I was tasked to come up with an approach to connect a restaurant chain's point of sale (POS) system to call our distribution center for order fulfillment. The problem was that this chain had a very old POS system, and it was not compatible with our software. So, what we did was build a simple loosely coupled configurable data transformation bridge that allowed us to connect those restaurant sites automatically with our distribution system without someone having to call in the orders manually. We soon then realized that this approach could be used for other chains on other types of point-of-sale systems.

2. PROBLEM

When an enterprise has multiple platforms and applications that manage a given workflow, it is difficult and complex to manage the end-to-end integrity and traceability of the transactions that drive that workflow.

This causes a proliferation of API's and complex integration channels. This, in turn, causes lack of context of transaction relationship to other transactions on the same workflow or context to the workflow itself.

3. FORCES

The issue lies in the integration, not individual the platforms or applications. Enterprise-level integration systems are complex and require significant design and maintenance. Accumulated technical debt, lack of data standardization and technical complexity make an enterprise integration architecture challenging to implement and maintain. Changes in source and target systems, especially during major business transformations, further complicate the issue by forcing unnecessary and constant application specific rework. This drives solutions that are application focused and not process focused.

Source transactions are often not in a format that target systems can receive, increasing data transformation complexity and reducing traceability and integrity. This requires complicated middleware solutions to bridge the gap between source and target systems. Configuration should be preferred over application specific programing in order to minimize application specific middleware solutions.

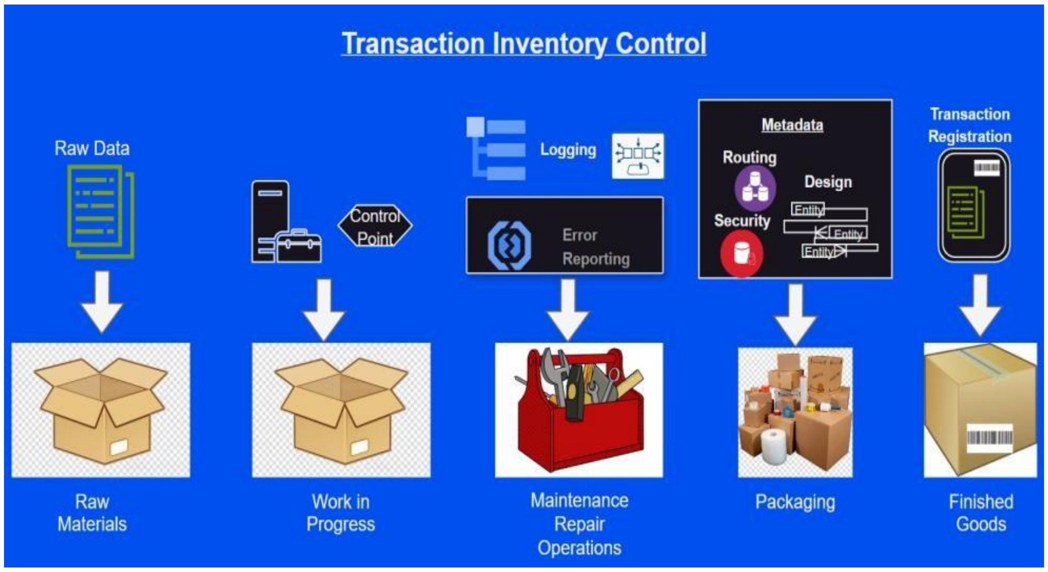
4. SOLUTION

What is needed is an application-agnostic and configurable solution that is adaptable and easy to maintain despite an ever-changing enterprise landscape, staying focused on the processes and data it supports.

Transaction identification and metadata management

Manage your data and transactions like product inventory in a supply chain to proactively identify, configure, and route them—regardless of source, destination, or content.

Transactions of a systems workflow is very much like products. Imagine your transactions are like products in an Amazon distribution center. The data of these transactions are the raw materials. The process, various control points, and structural design that a transaction goes through is the work in progress components. How you identify the error controls, maintenance logging can be considered the maintenance, repair, and operations of your inventory control. The metadata of each transaction is the packing materials. The primary packaging be the data model of the transaction itself. There is also the registration metadata that help organize the source target and data content control would be considering secondary packaging. The compartmentalization and security metadata that is include would be considered tertiary packaging to protect and insulate unwanted exposure. Finally, the finished transaction output is ultimately your finished set of products.



In inventory control systems, it is expected that you track, count, record, secure, and maintain the products that get routed through your distribution ecosystem. The barcoding and labeling of those products are crucial to the accuracy, timeliness, location, and security to the handling of those products through distribution. Systems transactions follow the same pattern. The proper barcoding and labeling of your transactional data out help ensure the accuracy, timeliness location, and security through the various control points of you transaction ecosystem.

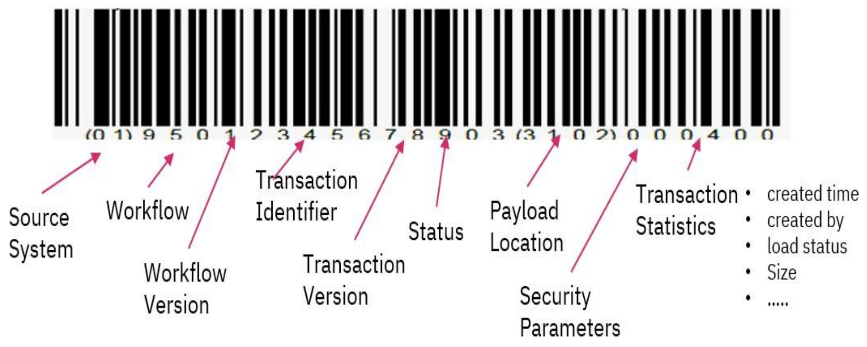
By building an integration architecture foundational based on the product inventory and distribution principles, allows the enterprise to focus and centralize the identification, security, routing of the transaction "products" that flow through an enterprise architecture. Your integration becomes more about the data and the processes it supports and no longer about the producer and consumer systems that are feeding that integration. It is this focus that will always allow your integration to evolve and scale regardless of technology changes.

5. IMPLEMENTATION

The implementation of this pattern is broken down into 3 areas, identification, workflow management and configurable processing.

6. IDENTIFICATION

Those two points in my past professional life, has provided me to treat enterprise data integration ecosystem, much like a product distribution network. In order to accomplish that, one must have a common approach to cataloging and identifying your data, much like product barcoding (Figure 1). The transaction barcode gives you a consistent way to identify and manage transaction in your distribution network regardless of the source. This is done by identifying components of that transaction and it's relationship to the workflow.



Transaction barcode
Figure 1
(reference 3)

Once the data is identified and catalogued, it needs to be maintained. Storage conditions and data retention must be assessed, much like product inventory. A system must be implemented to monitor the entry, storage, and exit. The auditability of the data and middleware processing is very much like the continuous monitoring and auditing of product inventory. If data integration is treated in the same manner, this allows for streamlined distribution through an event driven distribution architecture with event messaging, consistent meta data collection and common "barcode" like transaction identification (reference 1). This will allow for a streamlined and minimized persistence layer and a simplified reusable asynchronous API layer.

7. Workflow management

Every workflow has a typical start and end point. Once you have that concept, you can start to visualize what process and data output to that process triggers the instantiation of that workflow. This will then allow you to identify not only the workflow, but it will help to identify downstream transactions on that workflow. This is the most critical part of any data integration pattern (Figure 2).

If you have a common understanding that all transaction data is made up of data characteristics, then you can catalog those characteristics. Much like a blockchain's smart contract. How is that transaction identified in the system of record? This can be used to provide the ability to get additional data from the source system if needed later and to minimize what is sent through the integration layer. What are the minimum data entities and attributes that need to be captured for downstream processing activities? What is the format of the data as it is being passed into the integration layer?

A data pipeline is the production and processing of data from various data sources and then transformed for downstream consumption. This can be done with a small, centralized, configurable highly indexed relational data model design versus the large system specific designs of the past. Making the centralized product-like distribution highly scalable (figure 2), this approach can support multiple data pipeline approaches from batch processing to cloud native to streaming.

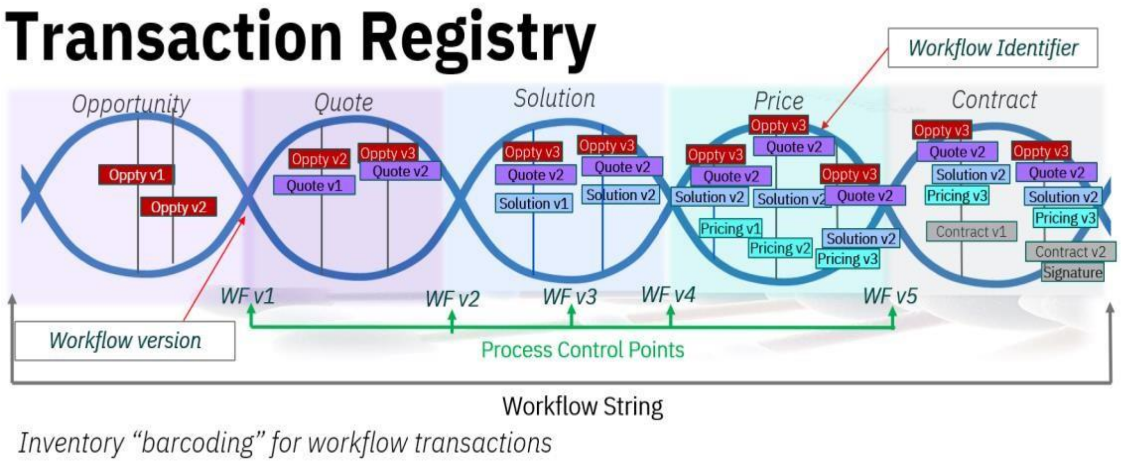
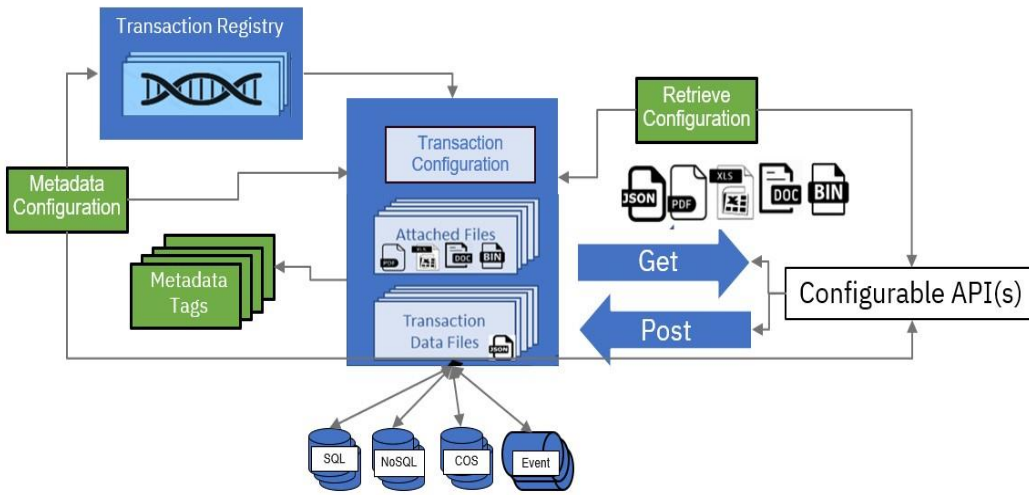


Figure 2

7.1 Configurable processing

Finally, by centralizing that transaction identification and metadata to include the workflow into a common barcode like approach, it allows for the following benefits. It provides a common mechanism for any endpoint to follow to participate, whether the endpoint is a producer or consumer. It greatly minimizes the customization that the integration layer must do regardless of end point. There is ultimately a single pipe entry in and out of the integration layer (reference 1). Centralizing allows you to communicate, route, and distribute the payload across multiple storage mechanisms (relational, NoSQL, document storage) tailored for the type of data passing through be it structured or unstructured. Through this centralized integration approach, it allows for a system to drive the access and security of the data at the registry or barcoding of the data thus greatly minimizing the attack area**.**



Integration architecture simplified
Figure 3

7.2 RESULTING CONTEXT Common reusable API layer

Once established how to treat the data content like product inventory no matter the system of record, it provides the opportunity to come up with a common reusable and configurable approach to our API development. Because we used a common metadata pattering and barcode-like transaction and workflow identification we only need one API to manage the processing of that data. We are now able to move away from system specific APIs to post transaction to the integration layer.

This pattern in turn could also be used to retrieve data using a single API to get the entire payload passed through integration or a configurable SQL based API that allowed for retrieval the highly indexed transaction catalog. The retrieval approach because of its configurability could then be linked through other sources greatly minimizing the amount of data that would need to pass through integration to satisfy a logical unit of work. This pattern also helps in minimizing the amount of data redundancy that is introduced. We no longer had to create custom APIs for various set of data passed through the integration layer.

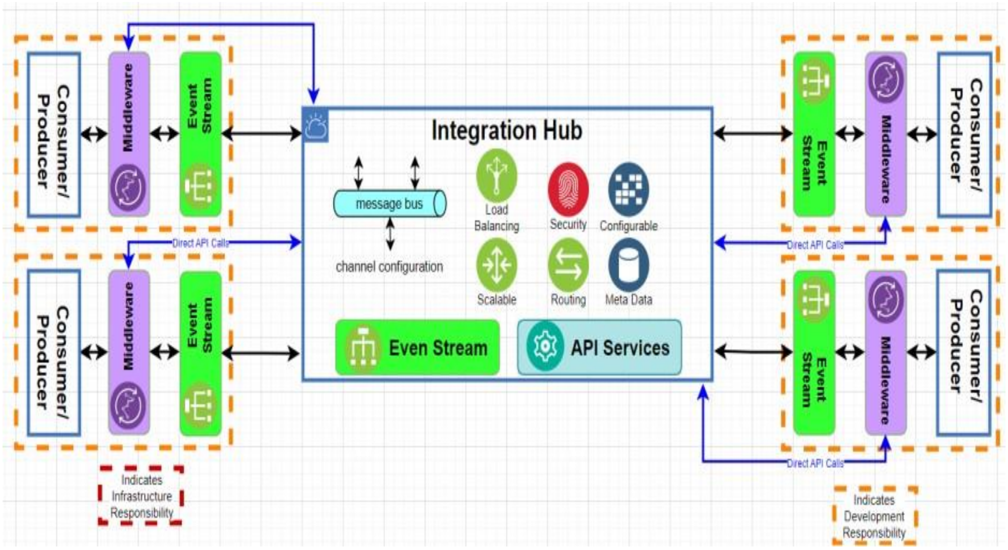
Treating your API management like I had to managed what tools I would put in my pack, provides a much leaner and more efficient development approach which in turn allows for a simpler scalability approach when more resources are needed. By building in configuration into the API layer itself it allowed for the creation of a single "post" API with metadata configuration and a single "get" API with retrieval configuration. These same APIs can be reused regardless of the source or target platform (figure 3).

8. EVENT MANAGEMENT AS MIDDLEWARE

A separate but related pattern is how event management can be infused into an inventory driven integration ecosystem.

Like transaction inventory pattern above equates to an Amazon distribution, the event management that equates to the shipping of your transaction products between the various source and targets.

Putting an **event broker** between the system endpoints and the central integration layer allows for the following benefits. This is very much like an inventory management system monitoring and tracking capabilities. First it allows for more loosely couple and asynchronous communication mechanism. It minimizes the amount of direct API calls that need to be made. It also minimizes the amount of data that needs to be persisted in the integration layer. Finally, it provides a transparent, end to end visibility of the entire distribution network where in the past most centralized integrations with middleware components the middle ware communication would be mask from the system end points. You need to treat your **event messages** like metadata as if that data was to be stored. You need to centralize and standardize the access and security of the event broker layer. And finally, the distribution and categorization of each **channel** is centrally configured from channel type (publish-subscribe, point to point, selective consumer, invalid message, …..) to producer and consumer routing (reference 1).



Centralized event management and choreography
Figure 4

A canonical data model (reference 1) can be created for the event layer. Every event created in the event layer should follow the same metadata patterns followed as discussed with the data that passes through the integration layer. The event headers should have a **common format** that describes source and identifies and categorizes the event message body, and retention characteristics. The event body should follow a common reusable pattern that can be validate prior to processing. This pattern should be followed for all messages, including acknowledgement and error responses. It is this consistency the simplifies the integration behavior of those messages both for participating end points and the central integration. This can be accomplished within the event layer with the use of AsyncAPI (reference 2).

Security and access to events should be centralized set up and managed. This simplifies the security footprint and management. Most of this is built into the event brokers natively. How it is configured needs to follow the same data design patterns of the data and events themselves.

Once you have a consistent configurable event message format and a centralized access pattern, then you can set up the varies channel distribution networks that need to be used. Much like a trucking routing system for a food distribution network. This is done by configuring the producers and consumers with the communication formats they use. The data formats they provide (JSON, XML, etc.) and how they will communicate with the event broker. Then the channel itself is configured with data transformation needed, focused microservice code specific to a channel needed for successful distribution and layered channel to channel ordering and connectivity.

9. SUMMARY

Logistics principles derived from my military and early career experiences have helped me streamline enterprise data integration architectures. Inspired by the necessity of multi-use with limited space and resource drives simplification and reusability in integration architectures regardless of size and complexity of the systems being integrated. Challenges with outdated point of sale systems using versatile, loosely coupled data transformation bridge allowed for the automation of the ordering process. This solution not only resolved the issue at hand but proved adaptable for other systems. This emphasizes the value of a flexible integration approach.

Drawing parallels between product barcoding in distribution networks and data management in IT systems highlights the importance of a standardized, cataloged approach to data handling that mirrors inventory management. This involves a common methodology for workflow identification, transaction metadata management, and a centralized, minimalistic integration layer that supports scalability and security while reducing redundancy and customization needs.

These strategies can be applied in various enterprise transformations, gradually shifting towards a more durable, system-agnostic framework. This shift was motivated by the need to avoid repetitive, costly redesigns with each system upgrade. This results in a streamlined, efficient integration system that minimizes complexity and enhances adaptability across different systems and data types.

Furthermore, the implementation of an event management middleware system mirrors inventory tracking, enabling asynchronous communication and reducing the reliance on direct

API calls. This centralizes and secures the integration process, akin to how a distribution network operates, ensuring efficient, transparent data flow across the enterprise.

In conclusion, by applying principles of simplicity and reusability, leads to a robust, scalable integration ecosystem that can adapt to changing technologies without requiring complete overhauls, thereby reducing costs, and improving operational efficiency.

ACKNOWLEDGEMENTS

I would like to thank Joseph Yoder, Eduardo Guerra and the rest of the PLOP committee for helping me refine my thoughts on this article. I would especially like to thank my mentor, Kyle Brown for the countless session and advise in helping bring this pattern to life and exposing me to this writing style.

REFERENCES

1. Enterprise Integration Patterns
Gregor Hohke, Bobby Woolf (2012). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison-Wesley Educational Publishers Inc, 2004.
(This foundational book details patterns for integration and messaging in enterprise systems, emphasizing simplicity, reusability, and scalability.)

2. Data Pipelines
AsyncAPI Specifications
AsyncAPI, accessed 15 April 2024, www.asyncapi.com.
(Useful for standardizing asynchronous communication, important for configurable transaction integration patterns.)

3. Types of Barcodes
International Barcodes, Published 2024, https://internationalbarcodes.com/types-of-barcodes/.
(Provides an analogy of barcode standards for managing transactions and their metadata, crucial for traceability and consistency.)