Domain-driven design and the future of payments

Thought Machine | Rethinking

Introduction

Author:

Yoav Ash
Senior Product Manager, Vault Payments
Thought Machine
yoav@thoughtmachine.net

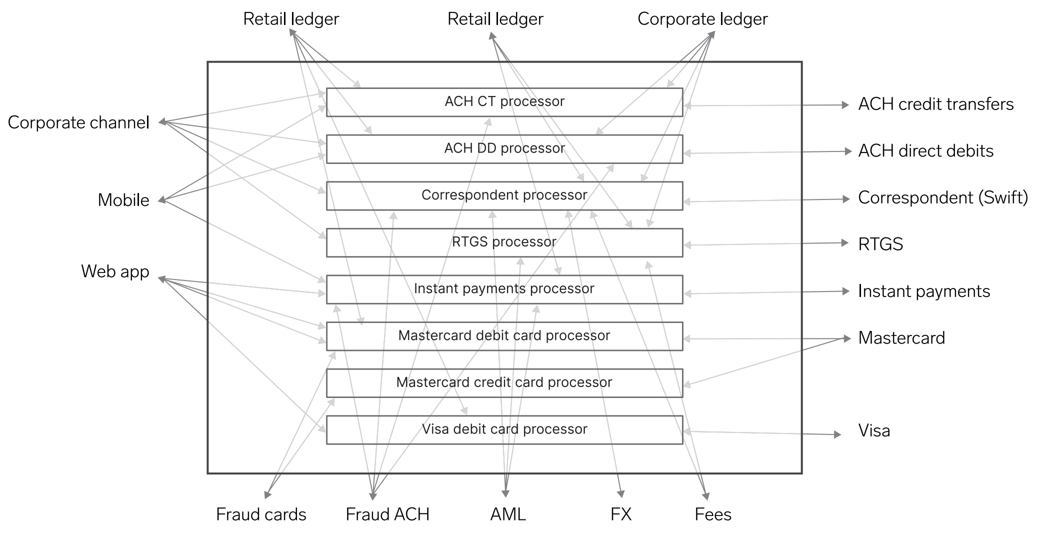
Over time, payment orchestration platforms in banks have evolved as a collection of isolated systems to serve the specific requirements of different payment rails.

As a result, aggregating front-end channel access, integrating with the bank stack, billing, and performing analytics across these systems can be difficult.
It is time and resource-consuming when changes are needed to comply with new regulatory requirements or to add new features to enhance customer experience. Supporting new payment rails or use cases often entails significant investment and business disruption. The need to operate multiple systems also comes with cost considerations, including licensing and operations support.

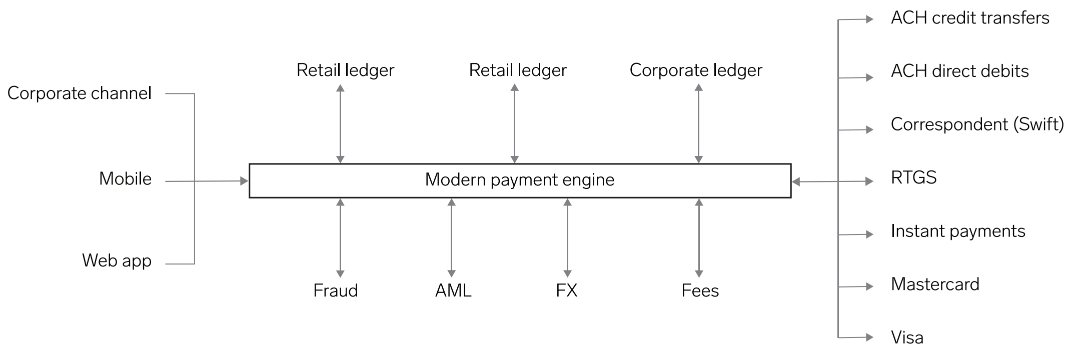
Why do banks need a new payment engine?

Banks face the challenge of handling multiple and diverse payment methods, including traditional methods such as cash, cheques and wire transfers, and newer methods such as digital payments, mobile payments, and e-commerce payments. Each payment method has its unique characteristics and requires different systems and processes to support it, and consequently, a common payment stack consists of multiple payment platforms.

It stands to reason that banks want to consolidate payment processes into a single platform. By doing so, banks gain operational and cost efficiency, have less exposure to cybersecurity and regulatory compliance risk, and can offer a frictionless customer experience.



What today looks like



Imagine a much cleaner picture

However, consolidating multiple and disparate payment methods, schemes, and types into a single platform presents challenges.

Different payment methods may have varying business requirements, compliance standards and procedures—leading to additional technical considerations. For example, a credit card processor must be able to respond to authorisation requests within a matter of a few hundred milliseconds. A credit card processor must also be aware of the card scheme's guarantees.

In contrast, the processing of direct debit batches is not time sensitive, but the processor must be able to schedule sending payments and settle them over multiple days. Modern instant payment systems such as RT1 and FedNow use single ISO 20022 messages, while older systems such as BACS and ACH use files with proprietary formats.

Banks won't find a payment system with connectivity to all payment methods because they cannot anticipate enough of the new payment methods that may arise in today's fast-evolving and complex financial world. A simple look at the payment methods that came onto the scene over the last two years makes the case.

And so here is the key point of this paper: Banks need a single payment orchestration system to simplify their stack and lower the cost of ownership and change, while gaining full control over the behaviour of their payments and the flexibility to future proof the solution, add new payment methods, and innovate.

“Banks need a single payment orchestration system to simplify their stack and lower the cost of ownership and change”

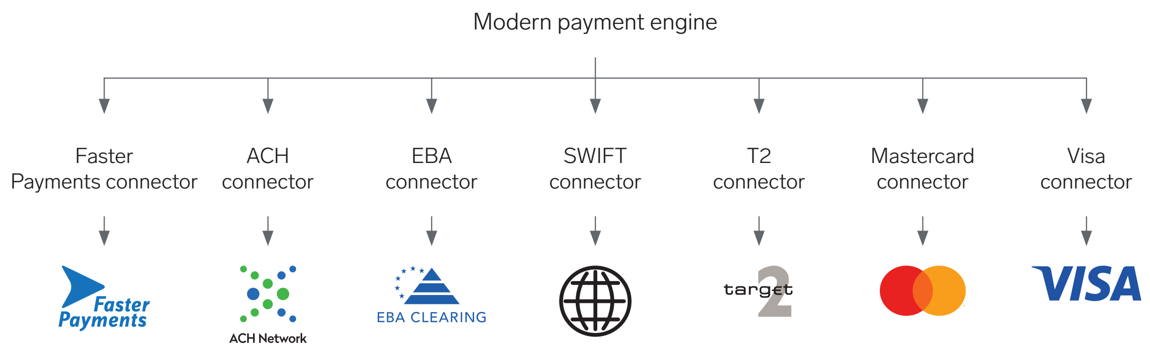
What should a modern payment engine look like?

Thin connectivity components with a universal configurable orchestration engine

Consolidation by simply combining everything in a single box is not possible because connecting to most payment systems still requires dedicated hardware and software, with limited or no ability to be shared across rails.

Therefore, it makes sense to maintain connectivity components dedicated to each payment system. These components should be as 'thin' as possible and be concerned solely with connectivity protocols and transformation so that common functionality can be provided by the engine and not duplicated in every payment system connector.

"These components should be as 'thin' as possible and be concerned solely with connectivity protocols and transformation so that common functionality can be provided by the engine"



An example of a universal configurable orchestration engine

Giving banks the ability to build their own payment orchestration

When it comes to orchestrating the life cycle of different payment types, the challenge is significant but not insurmountable. All payments are the same at their core: a movement of funds from one account to another. That being the case, do we still need a dedicated solution for each payment type? Would it be possible to build a solution that could support multiple flows, and furthermore, could it be possible to let users create these flows?

Before looking at how we have achieved this for payments, let's look at an example from a completely different domain: video games.

In the early era of game development, engineers had to build their technology from scratch to support capabilities such as physics simulation, rendering, connectivity, and scripting systems. This process was labour intensive and required significant programming expertise and a deep understanding of the underlying hardware.

Today, however, game engines like Unreal and Unity have changed how video games are made by providing developers with a pre-built framework for creating games. These engines include many features, such as physics simulation, rendering, and scripting. These features allow developers to spend more time focusing on their game logic, using scripts that utilise the underlying technology provided by the game engine.

The game engine is not a general-purpose software-building tool. It provides a specific set of tools aimed at game building, making it possible to build vastly different game genres because the fundamental tools they require are the same.

Games are resource intensive, which raises the question of whether a general-purpose engine can deliver the expected performance. In reality, since these engines are built and optimised by people who are experts in their field, they achieve far superior performance to what most developers could achieve if they were to build their own custom engines.

The approach described above applies to payments because it allows banks to build payment orchestration easily and quickly—you don't need an army of engineers to create everything from the ground up.

"These engines include many features, such as physics simulation, rendering, and scripting"

Domain awareness

One common aspect of payment processing is that it can be considered a production line, where a conveyor system moves payments from one workstation to the next. In the case of payments, which workstations to use, and what order to follow will change depending on the payment type, and our payment platform must enable users to configure their own payment production lines.

This might imply that the right tool for the job is a workflow engine, but in practice, this is far from the case.

To begin with, a payment engine must be highly performant, reliable, and dynamically prioritise payments based on their service level agreements—something that workflow engines fail at. Additionally, a payment engine must be payments-domain aware and provide specific tools that developers can use across payment types; otherwise, it will be left to users to build these tools.

For example, the basic steps of payment processing are validation, authorisation, clearing, and settlement. Each step has specific requirements and processes depending on the situation, but they all ultimately work towards the same goal of completing a successful transaction.

"To begin with, a payment engine must be highly performant, reliable, and dynamically prioritise payments based on their service level agreements—something that workflow engines fail at"

How should a modern payment engine work?

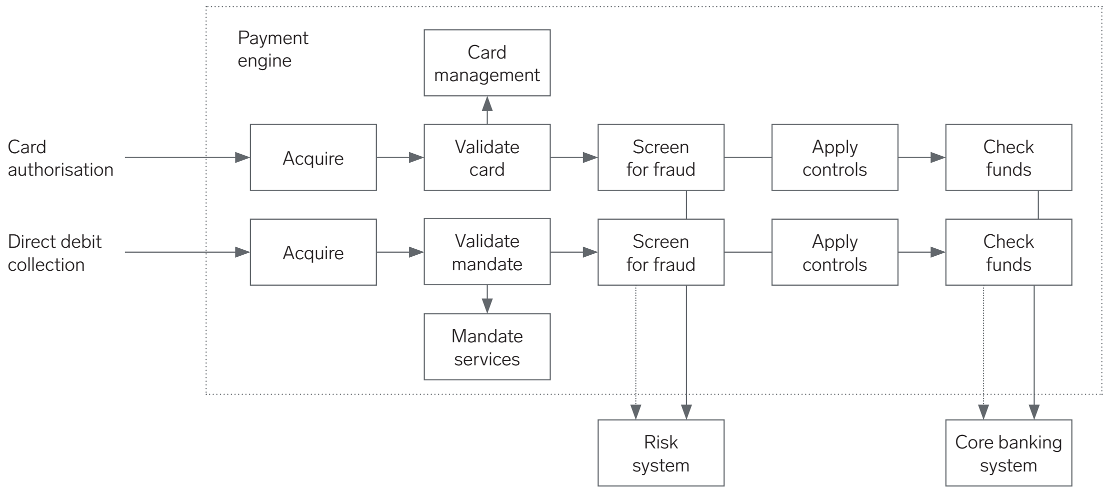
Let's return to the payment methods we covered in the previous examples: a card authorisation and a direct debit. In essence, both represent the intent of the customer to authorise a merchant to draw money from their account. So perhaps not surprisingly, the steps to handle them are quite similar:

1. Acquire the message from the payment network
2. Verify the transaction against a pre-existing authorisation data record
3. Consult a fraud engine to have some level of confidence that the customer indeed authorised the payment
4. Apply controls – limit the number or type of transactions
5. Ensure that the customer has enough funds in their account to cover the payment and reserve the funds for settlement

Looking at a credit transfer instruction from a customer to pay out of their account, we can still see similarities in the process. The payment must be screened for possible fraud, and the customer must have sufficient funds to cover the payment. This payment type requires other steps as well—to validate the destination of the payment, calculate processing dates and perhaps assign a fee.

The modern payment system should be a truly payment-type agnostic platform which is still payment-domain aware. It must represent all payment types using a common standard (ISO 20022 is now the de facto standard for payments worldwide), and it must encapsulate common payment processing steps into building blocks, allowing users to organise them in the correct order and inject them with user and payment system-specific logic.

All this must be delivered while keeping the payment platform performant and reliable. It is agnostic to the payment type, using the same engine to process them all, but able to prioritise based on the expected SLAs for each.



A workflow example of a modern payment engine

In summary

The payment landscape is rapidly changing, combining old and new methods, resulting in new customer expectations. Consumers now demand a seamless, convenient, and secure payment experience that they can easily access through various devices.

The emergence of fintech companies and non-bank payment service providers has added to this challenge. They often offer more innovative and customer-centric solutions, resulting in a shift away from traditional banking services.

To counter this trend, banks must prioritise technology investments to keep pace with the changing payments landscape and provide their customers with the level of convenience and security they expect. Failure to meet these expectations could lead to a loss of customers and a decline in the bank's market share.

Payment rails are increasingly commoditised and serve as mere connectivity mechanisms to communicate with specific payment schemes. Future payment systems' true power and flexibility lie in the orchestration engine that can effectively manage and streamline payment flows across multiple payment sources.

The orchestration engine is the brain behind the payment system, enabling businesses to offer their customers the best payment experience while optimising their payment operations.

An effective orchestration engine can allow businesses to customise payment flows, manage risk, and provide real-time payment insights.

To effectively evaluate a modern payment engine, banks should take a holistic approach and consider the tools available and their capacity to accommodate diverse use cases, as opposed to simply assessing the payment engine's compatibility with specific, known scenarios—adopt a forward-thinking mindset and extend beyond the limitations of existing offerings.

To enable the integration of new payment methods or foster interconnectivity between multiple modes of payment, it is sensible to seek out a payment engine equipped with the requisite resources: payment method agnostic, ISO 20022 compliant, performant, and reliable.

Finally, the ideal payment engine must deliver unparalleled configurability but not at the expense of performance and reliability.

About the author

Yoav Ash
Senior Product Manager, Vault Payments

Yoav Ash is an experienced product manager and solution architect with 20 years of expertise in fintech. He has worked for leading software vendors in the payment industry, facilitating large-scale transformations and greenfield projects.

Yoav possesses a deep knowledge of core banking, payment processing, and payment engines. He has successfully implemented various payment systems and schemes, such as SWIFT, TARGET2, SEPA, Faster Payments, and Mastercard.

About Thought Machine

Thought Machine has developed the foundations of modern banking with its cloud-native core banking and payments technology. Its cloud-native core banking platform, Vault Core, is trusted by leading banks and financial institutions worldwide, including Intesa Sanpaolo, ING Bank Śląski, Lloyds Banking Group, Standard Chartered, SEB, Lunar, Atom bank, Curve, and more.

Vault Payments is a cloud-native payments processing platform — comprising a Universal Payments Engine to support all card and account-to-account payment types.

Vault Core and Vault Payments have been written from scratch as an entirely cloud-native system and give banks full control to build any product required to flourish in a rapidly changing world.

Thought Machine is currently a team of 600 people spread across offices in London, New York, Singapore, and Sydney and has raised more than $500m in funding.

For more information,
visit thoughtmachine.net

To speak to a member of our team, email:
contact@thoughtmachine.net

Thought
Machine