How to design mixed payment methods with store credits

Scaling a convenient alternative to cash refunds sows the seeds for a multi-met payment system, and a sophisticated loyalty program

ALVARO DURAN
OCT 09, 2024

6 6

Most companies make a colossal mistake when they decide to accept alternative payment methods. They choose the wrong one to start with.

Once they accept credit cards, these companies' founders look outside. T see PayPal, Apple Pay. They want to give their customers more options.

And that's, to quote Bane from The Dark Knight Rises, fighting like a your man. "Admirable, but mistaken".

If you've never accepted payments beyond cards, there is a simple altern payment method to try. One that's completely under your control, but wi will give you the opportunity to build the scaffolding that will make integrating with the PayPals and the ApplePays way easier.

It all starts with how you refund.

I'm Alvaro Duran, and this is *The Payments Engineer Playbook*. Go spend 5 minutes on Youtube and you'll see tons of tutorials on how to pass softw design interviews that use payment systems.

But there's not much that teaches you how to build one for real users and money.

The reason I know this is because I've built and maintained payment syst for almost ten years. I've been able to see all types of interesting conversations about what works and what doesn't behind closed doors.

And I thought, "you know what? It's time we have these conversations in public".

In *The Payments Engineer Playbook*, we investigate the technology that transfers money. All to help you become a smarter, more skillful and mor successful payments engineer. And we do that by cutting off one sliver o and extract tactics from it.

Today, we're going to look closely at store credits, and how a protocol wi bad reputation is a great fit for mixed payments.

But first, I want to address the elephant in the room: why even bother wi credits?

Like I said, it's all about how you refund. When customers return what th bought, companies often give them the amount paid, in cash.

This is completely fine. But it's expensive, if they paid with card. For companies that sell products that are either low priced, or low margin, refunds **often cost a significant percentage of the profit**.

If you're refunding a card payment, you will need *a few sales to offset that* That puts an incredible amount of pressure on your company.

Most of the time, your only recourse is to be *continually selling at discou a non-refundable policy*. That's why it feels as if many fashion stores are always on "sale".

From a business point of view, offering a refund is **bad business**.

But you've probably noticed that well known fashion stores don't give cas refunds that easily. What they give you instead is a branded card, preloac with the refund amount.

This is called **store credits**. Or, in payments lingo, a *staged closed loop wa*

Store credits are amazing for three reasons. First, they **keep the money within the business**.

Second, there's the **breakage**, which is the industry term for funds unuse the customer. You get to keep the returned product, and some of the mo that was used to pay for it. Although depending on where you're doing business, the government might claim that money. Check this fact sheet Consumer Reports if you want to learn more.

And third, store credits are an **incentive for future purchases**. Every cor that implements store credits ends up realizing that the retention featur store credits overshadows the savings from not having to give cash back.

That last point only can explain why you come back to Zara or H&M mor often than to one of those struggling fashion stores that give cash refunc

Store Credits is an Async Ledger

How should you go about implementing store credits?

I've been working with store credits intensely for the last couple of week the biggest surprise for me was realizing that store credits is an asynchr payment method.

Which may be a bit surprising at first, but it makes total sense, because **s credits are meant to be used in combination with another payment me**

Yes, you can make a payment exclusively with store credits. But in practi
that's the exception, not the norm. In fact, your company benefits the m
from store credits *especially when they're used to generate more revenue*.

As a result, the best thing you can do from a software design perspective
lean on mixed payments with store credits from the very beginning.

Not only is it inevitable, it's also something that you might decide to
implement in the future, and *store credits are the simplest payment meth
with which to build mixed payments*.

Store credits, therefore, must have a **prepare** stage, where the amount o
store credits to be used is locked; and an **approve** stage, where the amou
removed from the account.

> *Notice that these are analogous to the authorize and capture stages of c
> payments. But I'd rather use terms that don't overlap with other domai
> because it muddles the conversation, and confuses the hell out of everyb*

In practice, this means that you need to keep track of credit transactions
could be in one of three states: *pending*, which are created during the pr
stage; *posted*, which result from the approve stage; and *discarded*, if you
to keep track of credit transactions that weren't approved (you should!).

This smells like a ledger, and it is. I strongly recommend that you also ke
track of the funds added into the users' accounts from the business, and
implement a complete ledger, rather than focusing on the users' side onl

I didn't track those funds once, and [it was terrible](#).

How to mix card payments with store credit

Async store credits let you mix store credits with card payments, becaus multiple async methods can be combined using a Two-phase commit (2P protocol.

If you've never heard of 2PC, it's a consensus protocol that coordinates tl work around a launch status check. The coordinator polls every node for go/no go decision, and commands every node to act only when all of the have replied with a go.

Now, I'm aware that 2PC has a bad reputation in the distributed systems community. Gregor Hohpe famously wrote that Starbucks doesn't use it. didn't mention that explicitly in my article on Sagas, but the Saga patterr the direct result of the frustrations that arise from using 2PC at scale.

2PC has three main downsides:

- **O(n) time complexity**: More nodes means more go/no go messages f around.
- **Single point of failure:** The protocol revolves around the coordinato crashes, even when all nodes have replied with a go, the combined ac won't happen.
- **Reduced throughput:** The combined result is dragged down by the slowest node.

These are very valid points. But consider this:

- **Payments that combine an increasing number of payment methods increasingly less frequent**. Card only is the most frequent option; m with credits, less so; and mixed with yet another one, very rare.

- **Payment systems should be resilient to crashes anyway.** Even if the is paying with a card alone, your payment system is still coordinating payment.
- **Third party payment methods are by definition the slowest.** Throug isn't effectively reduced by introducing store credits

In other words, the tradeoffs of 2PC have little effect when it's used to m payment methods. Despite its reputation, **I believe payment systems sh use 2PC to combine multiple payment methods**.

How does 2PC work in mixed payments

You can combine two (or more) payment methods using a design that fol these steps:

1. The user's store credits get *prepared* (locked, but not approved yet)
2. The payment by card gets authorized (not yet captured)
3. If the authorization succeeds, you can *approve* the use of the user's s credits, so that they cannot be used in the future anymore. If not, the user's store credits get unlocked and can be used in the future.
4. Finally, the payment by card gets captured.

Notice that step 2 can be as complex as needed. The strength of 2PC con from not putting any constraints on how complex each individual node c be. In fact, step 2 can be a Saga in and of itself.

What store credits introduce is the ability to switch cash refunds for top up the user's store credit account.

What is the downside to store credits? **Fraud**. Lots of fraud, actually. **The better the program, the more it attracts fraudsters**, because store cred unlike credit cards, have less controls built into them.

Their simplicity is at the same time virtue and sin.

> Entrepreneurs who run gift card marketplaces either go out of busine become the payment industry's most ruthlessly effective fraud busters
>
> — Patrick McKenzie, The Fraud Supply Chain

That said, one of the things that impacted me the most when doing resea on store credits is this: **they enable sophistication**.

Well designed, store credits are the stepping stone for a more "industrial retention program. You may start with a humble ledger with your users' accounts. But give it long enough, it can become a fully fledged loyalty program.

And, in time, it can become something similar to Target's RedCard, responsible for more than 20% of the company's revenue.

This is why the engineering of payments is so fascinating to me. There ar millions of avenues to explore. But if you build the right thing, and see to that it grows with the company, there's a direct link between software ar profit.

And that, to me, is what valuable software is all about.

This has been *The Payments Engineer Playbook*. I'll see you next week.

PS: Do you have 2 minutes?

Years ago, when I became interested in finance for the first time, a newsl like this didn't exist. I sure wish it had. I had been a paying subscriber for Stratechery right after I read it for the first time. I read all of Michael Lev books voraciously. Especially *Flash Boys*.

But none of that helped me build better money software. I learned a lot, y But I couldn't apply any of it.

With *The Payments Engineer Playbook*, I set out to write what I needed ba then. What would've made me a better payments engineer. What would'v me to avoid lots of costly mistakes along the way.

There are a few newsletters that many of us in the payments industry kn about. But they are focused on one of two things: system design job interviews, and soft skills.

And look, I get it! These are probably the only topics that are broadly applicable for software engineers. System design interviews are mostly t same, regardless if you're interviewing for one company or another, and skills are valuable everywhere.

But what's going to put you ahead is **doing your damned job right**.

If you build software that moves money around, being a paid subscriber The Playbook pays for itself.

You may know that this newsletter is going to become a paid publication 2025. Before then, you can pledge a subscription for a lower price, becau soon as the year starts, the price will increase.

If you know somebody who can benefit from this newsletter, can I ask yo forward this email to them?

Even if they can't afford it themselves, *most companies have a training bu for their employees* (ask your manager). These things move slowly though you need approval from your company, better hurry up before the windo closes.

Type your email... Subscribe

And if someone you respect shared this article with you, do me a favor a subscribe. Every week I feel I'm getting better at this. That means that m articles on how to build payment systems are probably yet to be written.

You can only find out if you subscribe to *The Payments Engineer Playbook* see you around.

← Previous Ne

Discussion about this post

Comments Restacks

Write a comment...

Oleksandr Gornostal Oct 13, 2024

♥ Liked by Alvaro Duran

And a second question: let's say a store credits ledger is implemented as a separate sub-s what challenges do you see in using eventual consistency in the purchase flow?

One approach might be logging a payment intent in the database and processing the trans asynchronously. If something fails, appropriate rollbacks would be applied. Kind of like a S orchestration, but without extra complexity: without changing the downstream services or Saga middlewares.

♡ LIKE (1) REPLY

4 replies by Alvaro Duran and others

Oleksandr Gornostal Oct 13, 2024

Liked by Alvaro Duran

Thanks for the interesting read, as always!

I'm curious why a payment system isn't designed to avoid the need for complex distribute transactions? Such an approach could enhance cohesion and maintainability. A ledger wit credits could be implemented in the same service, as payments and store credits are inhe linked and share similar architectural characteristics. By separating them into distinct enti might gain no real advantage. Am I missing something?

LIKE (1) REPLY

4 more comments...