

The Clover Mini Contactless Performance Story — Part 1



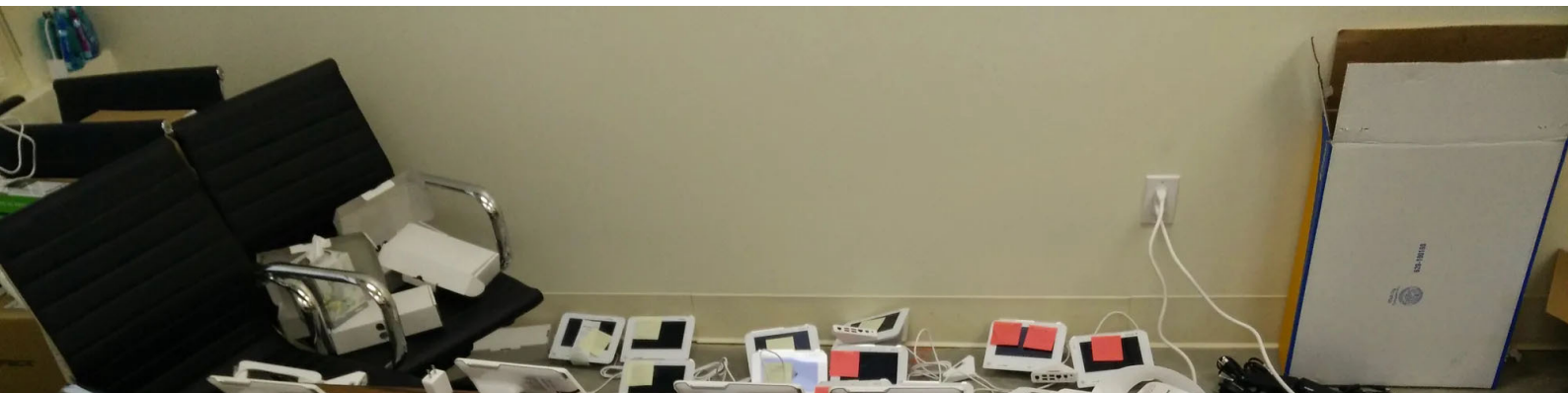
Clover Platform · [Follow](#)

Published in Clover Platform Blog · 4 min read · Jun 4, 2020



by Jacob Abrams

This story covers just a fraction of the journey we set on when we decided to build our first EMV-compliant, NFC contactless-enabled payment devices: Clover Mini and Clover Mobile.



Open in app [↗](#)

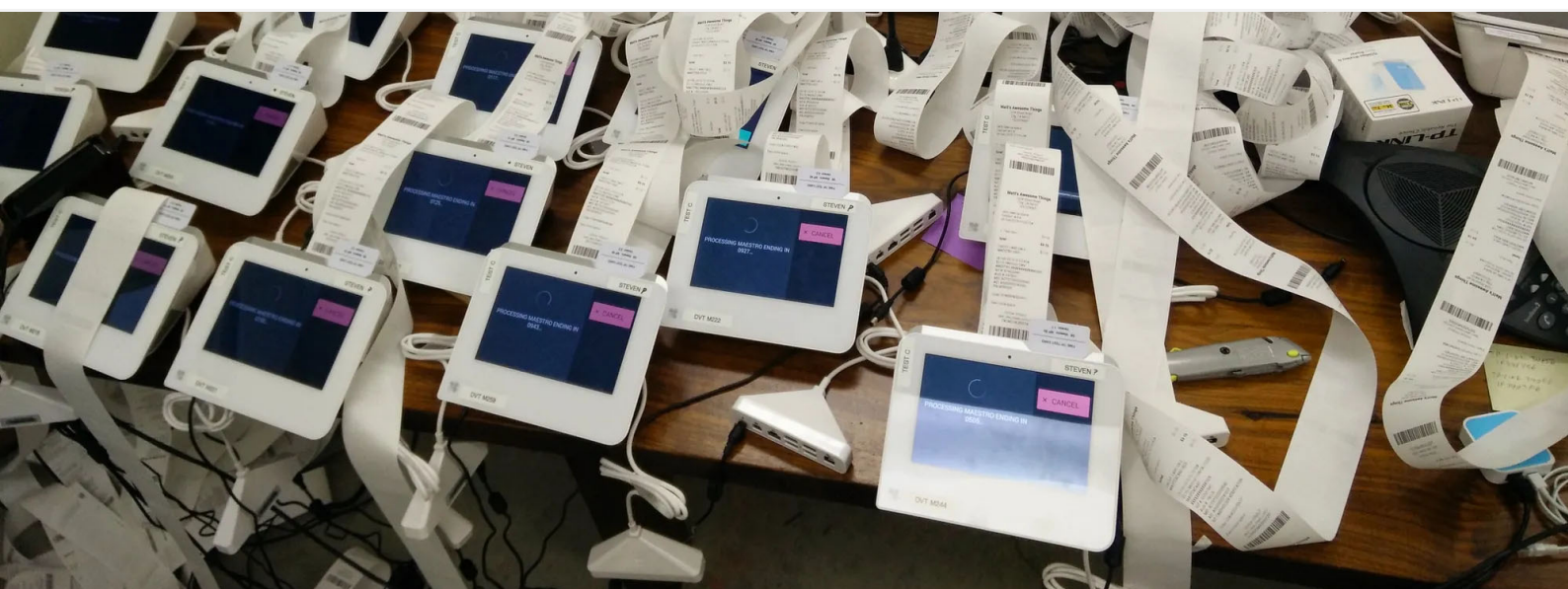
[Sign up](#)

[Sign in](#)

 Medium

 Search

 Write



Faster transactions equals happier merchants and customers

Payment terminals are required to pass a long list of certification tests before they land in the hands of merchants. MasterCard in particular has very strict performance requirements for contactless (aka NFC) transactions. MasterCard requires the total time spent by the payment terminal to be less than 100 milliseconds from the transmission of the first contactless packet to the last packet and audio tone indicating completion. They want to minimize the amount of time consumers have to hold their card or phone next to the payment terminal before the payment is complete. Faster transactions equals happier merchants and customers. Since payment terminal manufacturers such as Clover cannot control the time spent by the card or phone during the transaction, that time is excluded from the measurements

during certification. Our initial contactless test results for Clover Mini and Mobile came in at over 300 milliseconds, nowhere near the 100 millisecond requirement.

Beep latency

For security reasons Clover uses a separate secure processor to handle all sensitive card data. This secure processor communicates with the card during a contactless transaction but it has no display or speaker itself; our primary application processor controls the display and speaker. The two processors send each other messages over USB to complete a payment.

We began working on improving the performance of contactless payments by adding tracing to all the significant events that take place during a contactless payment. We noticed that there was a significant delay from the end of the transaction to actually hearing the beep come out the speaker. This audio delay was over 100 milliseconds in and of itself.

My coworkers began working on the problem and they made one critical change to help bring the time down. They switched from using USB messages between the processors to using an available GPIO we had connected between the processors to send the signal that it was time to play the contactless beep. The GPIO generated an interrupt that was received much more quickly than the USB message. However, the time between invoking the Android Java play sound function and the actual emitting of the sound from the speaker was still measured to be at least 85 milliseconds. We had to find a way to reduce the audio latency in the Android OS. One coworker suggested an interesting idea of starting to play the sound before it was needed but at zero volume and then increasing the volume to an audible level at the moment the beep sound was needed. I gave it a try but the performance improvement was negligible.

Circumventing the Android Java APIs

At this point I guessed that using the Android Java APIs to play sounds might be the source of the delay. Operating systems are composed of many parts and the Android OS is a combination of the linux kernel with a lot of user-mode code on top that provide easier to use Java APIs for app developers. Using the Android Java APIs to play sounds involves sending interprocess messages from your app process to the AudioFlinger in a system process which eventually processes the audio and sends it to the kernel audio driver. I decided to investigate how native Linux apps play sound since Android is based on Linux. After some searching I read about the ALSA API and then discovered in the AOSP external projects an open source TinyALSA app that used the ALSA API to play sound natively. The test results were promising. There was very low latency when using the ALSA API to play sound. I proceeded to build a JNI interface between the Java Android app I had built to manage secure processor communication to C code which used the ALSA API based on TinyALSA. The results were impressive — the audio latency was down from 85 milliseconds to about 7 milliseconds.

Even with this new solution, I noticed rare instances of a large delay of hundreds of milliseconds to play the sound. From playing with the device it soon became apparent that if any other sound was played by Android recently that would affect the performance of the native ALSA sound player. I searched through the AudioFlinger code for keywords like sleep, delay, and milliseconds. I discovered that a standby time of 3 seconds was used to retain a lock on the audio driver so that AudioFlinger need not re-initialize the driver. Reducing this lock time to a couple hundred milliseconds eliminated the contention. This change would, of course, affect the audio performance of audio intensive Android applications like games, but since our device was not meant for gaming the drawbacks were minimal.

...And yet so far from 100 ms

While we were able to save significant milliseconds by reducing the contactless beep delay, our actions were not nearly enough to get us down to 100 ms. Most of the code running on the secure processor during contactless

transactions was in the various software libraries that Clover didn't author but had assembled from a variety of vendors and open source projects. This meant we were confronted with analyzing large bodies of code we were not particularly familiar with.

The story doesn't end here... In part 2 of this story, I describe how we faced this challenge as we reached near completion on building the Clover Mini and Clover Mobile.

[Clover Network](#)[Payment Terminal](#)[Nfc Transactions](#)[Clover Mini](#)[Clover Mobile](#)

Written by Clover Platform

116 Followers · Editor for Clover Platform Blog

[Follow](#)

More from Clover Platform and Clover Platform Blog

Manage Environments

Environment Templates

Edit Environment

Clover Test Merchant (Sandbox)

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> url	https://apisandbox.dev.clover.com	
<input checked="" type="checkbox"/> mid	ESQPKVMKA4QZ0	
<input checked="" type="checkbox"/> api_token	4a06d3ec-fd38-5a7c-9bcb-033b22864455	
New key	Value	

Cancel

Update



 Clover Platform in Clover Platform Blog

Getting Started with the Clover REST API using Postman

We have a new version of this blog post!
Clover has made some improvements to the...

4 min read · Jun 22, 2017

 189  3 



 Bryan Vargas in Clover Platform Blog

Understanding Clover Auth Tokens and Ecommerce Keys

You may experience confusion about how to use tokens and keys access merchant data....

6 min read · Jan 12, 2023

 9  

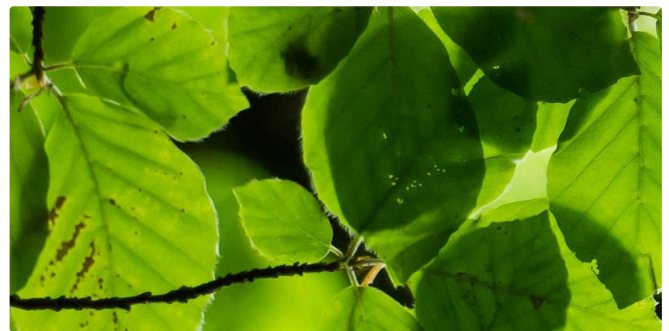
 Rachel Church in Clover Platform Blog

Modular SCSS and Why You Need It

In one of my most recent projects, we didn't have any global CSS classes. We didn't apply...

11 min read · Mar 28, 2019

 634  2 




 Clover Platform in Clover Platform Blog

Building a Microservice with Spring Boot and Spring Cloud

When we decided to move to a microservice architecture at Clover, different designs cam...

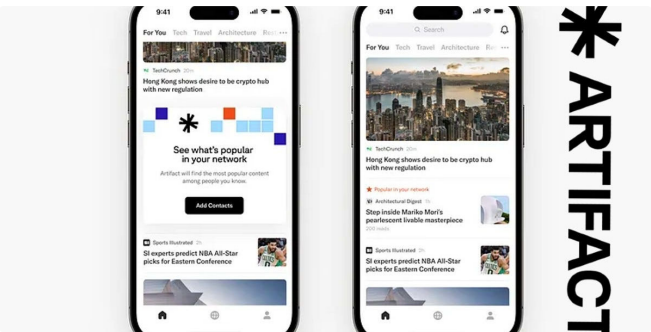
8 min read · Dec 13, 2018


 488  5 

See all from Clover Platform

See all from Clover Platform Blog

Recommended from Medium



 Gowtham Oleti


Apps I Use And Why You Should Too.

Let's skip past the usual suspects like YouTube, WhatsApp and Instagram. I want t...

10 min read · Nov 14, 2023

 13.5K  216



 Alexandru Lazar in ILLUMINATION

Ten Habits that will get you ahead of 99% of People

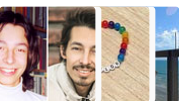
Improve your life and get ahead of your peers in 10 simple steps

9 min read · Nov 18, 2023

 16.5K  283



Lists



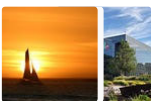
Staff Picks

552 stories · 617 saves



Self-Improvement 101

20 stories · 1170 saves



Stories to Help You Level-Up at Work

19 stories · 407 saves



Productivity 101

20 stories · 1072 saves



Companies have found that "Agile", as it is sold, delivered, and explained to them, does not work. You can blame them if you like, or you can blame the Agile community for not packaging the right kinds of learning and support. But regardless, "Agile" as we know it is dead. And Scrum will go with it.

But companies still need _agility_. Real agility. That has been our focus.

Real agility is mostly behavioral, and in particular, it is driven by the behaviors of leaders. Leadership is the big glaring hole in the Agile Manifesto. It is like trying to make concrete without water. No wonder "Agile" did not work.

That's why Agile 2, which reimagined what "Agile" should have been,

 Kurtis Pykes in The Startup

Don't Just Set Goals. Build Systems

The Secret To Happiness And Achieving More

★ · 9 min read · Dec 20, 2022



21K



327



Scott Galloway 

2024 Predictions

Each year, we review/make predictions re the past/coming year. Most years, we hit more...

11 min read · 3 days ago



3K



47



 Tamás Polgár in Developer rants

Agile has failed. Officially.

Either I'm a gifted oracle, and all of my friends are, or Agile really was just a stupid idea to...

2 min read · Dec 2, 2023



12.2K



379



Sheila Teo in Towards Data Science

How I Won Singapore's GPT-4 Prompt Engineering Competition

A deep dive into the strategies I learned for harnessing the power of Large Language...

★ · 24 min read · Dec 28, 2023



4.4K



47



See more recommendations