FreePay: An open standard for in-person crypto payments

v0.2

Abstract

FreePay is an open standard for in-person crypto payments. It is a set of specifications and software that will allow any user with any crypto wallet on any chain to pay for goods and services with a single NFC tap at any participating merchant.

FreePay is as free as possible being MIT licensed open source, having zero fees or middlemen, and enforcing no restrictions on the end user of this software.

Motivation

Despite payments being the earliest use-case for crypto, in-person payments has struggled to take off, despite many attempts. I think a large reason for this is the problem was being tackled by many small startups each in their own way with their own tradeoffs and lock-in to certain chains/tokens/wallets.

There is little chance of taking on the billion dollar incumbents with a set of small startups if these startups don't even work together.

FreePay is an attempt to create a set of open standards, software, and hardware, that are completely wallet/chain/token agnostic. This way any company working in the crypto payments space can build towards a common specification instead of reinventing the wheel time and time again.

In Person Payments

Where traditional payments fail

Legacy payment systems incur high costs, slow settlement times, chargeback risks, and require layers of manual setup and authorization.

From a customer perspective in-person payments are fast and efficient, however from a merchants perspective they are a costly and cumbersome exercise. Credit card payments can take days to finalize and remain subject to reversal for months. They also charge high fees, often up to 3 - 5% of the cost of an item, which could be over 50% of the margin.

Additionally, traditional systems present significant barriers to financial inclusion, leaving approximately 1.4 billion people unbanked and without access.

Where past attempts at crypto payments failed

There have been many attempts at real world crypto payments previously, most of which have failed. Let’s cover some of the reasons why, which we should attempt to avoid if we want to succeed.

Building a payment terminal and wallet that only work together - this is easier to implement, but trying to distribute this is vastly more work. You have to first onboard merchants without a customer base, then onboard customers to something new too. Even if it does work they're locked into a single network and company/app, which is hardly better than using the existing networks.

An even worse version of this is **using a custom token for payments**.

Locking to one chain - Crypto has become huge and with it there have been an explosion of chains. Many people have funds on a number of chains and want to use their favorite chain when shopping. This doesn't mean you need to support all chains - Just Bitcoin, Ethereum and Solana would probably cover 95% of the crypto space. But locking to just one is like being a company that only accepts Visa and everyone with Mastercard is unable to pay.

Crypto's unique strengths

There are a few areas that I think crypto really excels at that we should be embracing:

There are hundreds of wallets already, with hundreds of millions of users. We can leverage this and allow people to pay with what they already have rather than trying to introduce yet another thing.

Open Source and Open Standards are our friends - Crypto is all about open source, open access, open networks, the more open we can be the more support we get from those who believe in freedom and opportunity (which is most people).

Stablecoins make everything much easier to understand. Not only do people generally trust and like them more than crypto dollars, because crypto is such a great remittance device many people in developing countries already have them and can start their own local economies using the coins and wallets they already have.

How FreePay works

FreePay currently consists of a Merchant Terminal application (available in Android or NodeJS) and a Customer App (Android only). The customer app essentially acts as a bridge between the user's wallet and the merchant terminal. In the future wallets can adopt the FreePay specifications themselves and then the user won't need the FreePay customer app.

To use FreePay a merchant must already have a crypto address, this could be a self-hosted wallet they control, or an exchange. They initialize their device with this address as well as an Alchemy API key (which is free). They can optionally choose a set of chains and token addresses they will accept, by default it accepts all supported tokens on all supported chains.

When charging a customer the merchant first enters the total amount requested and clicks charge. The customer then taps their phone to the terminal. In the background the merchant's wallet requests the customer’s address, looks up what tokens they hold, determines the best token for payment and sends the user a payment request for that token in the exact amount they need. The user confirms payment and once the payment hits the chain the merchant's terminal will appear with a green "approved" screen (Fig 1)

The best token is chosen by the following algorithm:
- Stablecoins > Base Currency
- Cheaper Chains > Expensive Chains

Stablecoins are always prioritized as they have the most bridging options, cheapest fees, and most liquidity, and most merchants prefer being paid in a stable currency. If they are not available ETH or BTC can be used and it can be auto swapped to stablecoins if a merchant desires and sets their terminal up in that way.

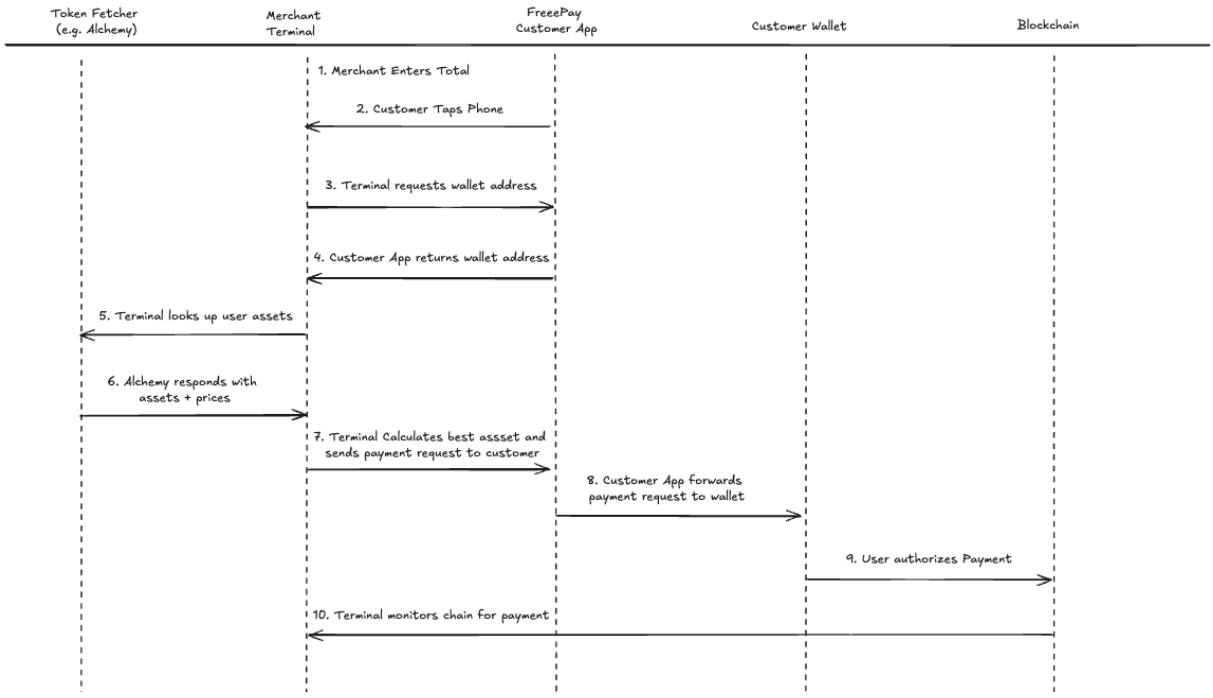

Fig 1

It is possible for wallets to integrate with FreePay directly, so their users don't need to install the customer app to use FreePay payment terminals. When this is done the payment flow looks like (Fig 2)

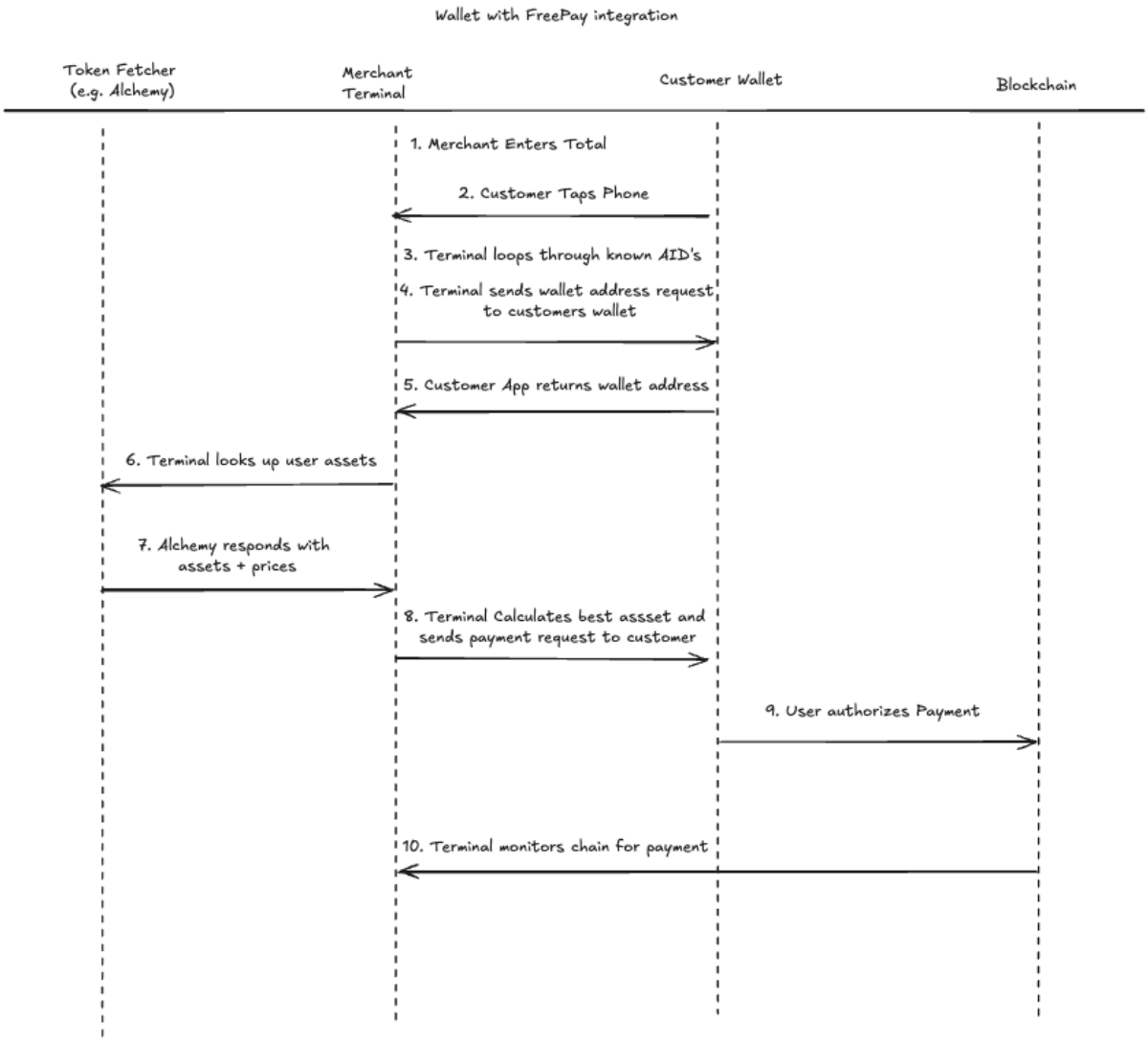

Fig 2

Each wallet will have its own unique AID (Application Identifier), this is a requirement of NFC tap to pay systems. FreePay will loop through all known AID's to find what wallet the user has installed, then proceed through the standard process of requesting a wallet address and sending a payment request exactly as it would with the FreePay customer app.

The reason the terminal requests the users assets before sending a payment link is to reduce friction, so the customer and merchant don’t have to manually negotiate tokens and chains, that is all handled in software.

The benefits over Traditional Finance

Merchant Benefits

FreePay's primary beneficiaries are merchants. For the longest time they've been straddled with high fees, long settlement times, and the possibility of malicious chargebacks. Many merchants are completely unable to get a credit card facility because they work in domains the big credit card companies consider undesirable.

- Payments settle instantly onchain, eliminating chargebacks and disputes.
- No reliance on banks or third-party approvals, ensuring global, permissionless access.

By removing these roadblocks we unlock finance for the billions of unbanked businesses and allow them to keep more of what they make.

Customer Benefits

Customers also have many benefits. They are able to hold the currency of their choice and have it auto converted at payment time, instead of needing to always hold local cash. They are in complete control of their money with zero risk of being de-banked, and can spend it wherever they like, similar to cash. Merchants may also give more rewards or discounts to customers paying with FreePay because it saves them so much in processing fees.

Micropayments made possible

Because of the zero fee overhead and tiny blockchain fees FreePay opens up many new possibilities for micro transactions. Now a hardware store is able to sell individual washers or fittings for a few cents, and many of the small farmers markets and fruit stands especially in developing countries are able to sell small amounts of items, without worrying about their payment infrastructure overhead.

FreePay has all the benefits of cash for merchants and consumers, but is available globally and with your choice of currency.

Distribution

This whitepaper won't go into too much detail on distribution, as the aim of FreePay is to provide that open source, free core that others can build upon. It could be taken to market by a DAO, a corporation, or another type or organization.

What is crypto's strength here? There is an army of profit seekers in crypto, let's get them distributing something the world wants.

A DePIN network would be a good model to copy. Many have achieved a supply surplus from token incentives, and are waiting for demand to show up. Now we have a payment device that has some demand but we need more of the supply.

The simplest way would be a referral scheme, though it would require some tracking so needs to be done by some semi-centralized org. Users could sign up merchants and in return they get:

- Some cut of the merchants fees (if this org decides to add fees on top of the device)
- Token incentives
- Future Airdrops

If there was some small fee it would make distribution far easier because users know every merchant they signup is a passive income stream built on real transactions. The token incentives and airdrops would be the cherry on top.

Prior to scaling up distribution, FreePay will focus on setting standards and ensuring all the most commonly used wallets support FreePay payments. It makes sense to do this first as there is little downside to wallets implementing these standards, only developer time. There is a large downside to scaling too early before wallets are ready as users will experience poor UX and may never try it again.

Appendix

Technical Specifications

Initial connection

When the customer taps their phone the merchant terminal searches for an application with a known AID on the customer's phone. The first AID it tries is `F046524545504159` which is `F0` + FREEPAY in Ascii -> Hexidecimal. If this is found it will connect to the application and proceed. If this is not found it will cycle through other known AID's for other wallets that have integrated with FreePay (currently this is 0). After it has found an application with a known AID it proceeds to the next step.

Requesting Customer Address

FreePay requests the customers address with the following NDEF message:

```
None
wallet:address
```

When the application receives this message it responds with a single CAIP-10 formatted address. e.g.

```
None
eip155:1:0xabc123abc123abc123abc123abc123abc123abc1
```

There is a CAIP (Cross-Chain improvement proposal) pending to formalize this request and make it a standard for wallets to provide their address.

Payment Request

The payment request format will differ depending on the chain supplied in the response to the getAccounts query. For EVM chains a ERC-681 [1] request will be sent. For Bitcoin a BIP-21 request will be sent. For other chains the equivalent method that makes the wallet open a send payment screen will be sent.

The request will be formatted as an NDEF string and sent to the phone. An ERC-681 request looks like this:

```
None
ethereum:0x833589fcd6edb6e08f4c7c32d4f71b54bda02913@8543/transfer
?address=0xabc123abc123abc123abc123abc123abc123abc1&uint256=20000
```

This is a request for $0.02 of USDC on Base chain paid to the address `0xabc123abc123abc123abc123abc123abc123abc1`

A BIP-21 request looks like this:

```
None
bitcoin:175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W?amount=0.001
```

This is a request for 0.001 Bitcoin paid to the address 175tWpb8K1S7NmH4Zx6rewF9WQrcZv245W

User Payment Action

After the payment request has been sent to the customer’s phone their chosen wallet should open with the destination address and amount already prefilled. They only need to confirm payment often using biometrics or face recognition and the funds will be sent to the merchant.

Technical Choices

Dealing with Cross-chain Complexities

The biggest problem with onchain payments is the plethora of chains and tokens out there. To get the user and merchant on the same page requires the use of DEX's and Bridges. There are two ways this could be implemented:

- Build this into each wallet, so the wallet can get a payment request and figures out on its own how to get its token to the correct destination chain/token.
- Build this into the terminal, so the terminal receives a payment of any chain-token and figures out how to get it to what the merchant wants.

Both are viable however based on how slow wallets are on upgrades and just how many of them there are, FreePay is opting to build all bridging and swapping into the terminal itself. Currently the LayerSwap bridge has been implemented which supports 35+ chains and most common stablecoin or base token sending between them.

Future Proofing

FreePay has been designed to be token/chain/wallet agnostic so is able to be upgraded to support new payment systems into the future.

As the FreePay Merchant app is an Android app it can be updated over the air by merchant terminals as support for new wallets, chains, or tokens is required.

Integration Examples

Wallet ERC-681 implementation

For a wallet to work with the FreePay Customer app it only needs to add ERC-681 intent support. Once FreePay supports other non-EVM chains the wallet will need to support the equivalent transaction request for that chain.

After ERC-681 support has been added the wallet will need to be added to the auto detect list inside the FreePay customer app. This is a list of all known wallets that FreePay detects and shows to the customer when they first install the customer app.

Currently only 2 wallets support ERC-681 intents:

- Metamask
- Base

Wallet NFC Tap Implementation

A wallet can go further and implement tap to pay directly. This will allow customers to pay with that wallet without needing to download the FreePay customer app at all. To do this a wallet must first create their own custom AID beginning with F0 (FreePay's AID is F0 + FREEPAY in HEX format).

They must then implement a response to a `wallet:address` NDEF message where they will return an address in CAIP-10 format.

After this is completed they must submit a PR to the FreePay merchant app to add their AID to the list of known AID's, so that the terminal can detect this wallet and work with it.

References

[1] ERC-681 Specification - https://eips.ethereum.org/EIPS/eip-681